UNIVERSIDADE DE CAXIAS DO SUL
CENTRO DE COMPUTAÇÃO E TECNOLOGIA DA INFORMAÇÃO
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

JOSÉ LUÍS NICOLETTI

Integração de Bancos de Dados Heterogêneos Através de Consultas Globais

Prof. Dra. Helena Graziottin Ribeiro
Orientador

Caxias do Sul, Dezembro de 2009

"Nosso cérebro é o melhor brinquedo já criado:
nele se encontram todos os segredos,
inclusive o da felicidade."
— Charles Chaplin

AGRADECIMENTOS

Antes de tudo, agradeço a Deus, por ter me dado a oportunidade de chegar até essa etapa da realização de um sonho.

Agradeço aos meus pais pelos ensinamentos, educação, orientação, apoio e carinho durante todos esses anos dessa longa caminhada. Agradeço, também às minhas irmãs, pelo apoio recebido durante esta jornada.

A todos os meus amigos que, distantes ou não, torceram pela realização deste trabalho e pela realização de todos os meus sonhos.

Aos meus colegas de curso e de trabalho, companheiros de tantas alegrias e preocupações, que estiveram comigo nestes últimos anos da minha vida.

Agradeço aos professores que auxiliaram na minha formação, em especial à minha orientadora, Prof. Dra. Helena Graziottin Ribeiro, pelos ensinamentos, paciência, dedicação e pela tranqüilidade que sempre me passou.

Enfim, gostaria de agradecer a todos que de alguma forma contribuiram para que esse momento fosse possível. Obrigado a todos!

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS 7

LISTA DE FIGURAS 8

LISTA DE TABELAS 10

RESUMO 11

ABSTRACT 12

1 INTRODUÇÃO 13

2 SISTEMAS DE BANCOS DE DADOS HETEROGÊNEOS 19

2.1 Métodos de Integração de Bancos de Dados 20

2.1.1 Bancos de Dados Federados 20

2.1.2 Armazéns de Dados 21

2.1.3 Mediação 23

2.2 Problemas de Integração de Informações 23

2.2.1 Distribuição dos Dados 24

2.2.2 Autonomia dos SGBDs 25

2.2.3 Interoperabilidade 25

2.2.4 Diferenças nos tipos de dados 25

2.2.5 Diferenças nos valores 25

2.2.6 Diferencas na semântica 26

2.2.7 Omissão de valores 26

2.3 Abordagens para Integração de Bancos de Dados Heterogêneos 26

2.3.1 Integração Baseada em Esquemas 26

2.3.2 Integração Baseada em Transações 30

2.3.3 Integração Baseada em Linguagem de Consulta 33

2.4 Considerações Finais 34

2.4.1 Vantagens 34

2.4.2 Desvantagens 35

3 MEDIADORES 36
3.1 Projetos desenvolvidos utilizando mediadores 38
3.1.1 HERMES (*Heterogeneous Reasoning and Mediator System*) 38
3.1.2 J-Integrator 38
3.2 Arquitetura de mediação em três níveis 39
3.2.1 Operações e Gerenciamento do Volume de Dados em Mediadores . . . 40
3.2.2 Conversores, Adaptadores ou *Wrappers* 42
3.2.3 Funcionamento da Camada de Mediação 44

4 PROCESSAMENTO DE CONSULTAS GLOBAIS 46
4.1 Processamento de consultas em SGBDHs na visão de (MENG; CLEMENT, 1995) 47
4.1.1 Decomposição da consulta 48
4.1.2 Tradução da consulta 48
4.1.3 Otimização global da consulta 48
4.2 Processamento de consultas em SGBDHs na visão de (LITWIN; MARK; ROUSSOPOULOS, 1990) 49
4.2.1 Formulação de consultas 49
4.2.2 Transformação de comandos 49
4.2.3 Processamento e otimização das consultas 49
4.3 Comparação das abordagens de (MENG; CLEMENT, 1995) e (LITWIN; MARK; ROUSSOPOULOS, 1990) 50
4.4 Processamento de consultas com o uso de mediadores 50
4.5 Arquitetura do processador de consulta 51
4.5.1 Tradutor de consultas 51
4.5.2 Processador de consultas 51
4.6 Exemplo de processamento de consultas 52

5 PROPOSTA DE ARQUITETURA DE MEDIADOR BASEADO EM CONSULTAS GLOBAIS 54
5.1 Escopo do mediador 54
5.2 Comandos de consulta a serem abordados 56
5.3 Processamento de consultas 59
5.3.1 Identificação dos elementos da consulta 59
5.3.2 Localização dos dados 59
5.3.3 Decomposição da consulta 59
5.3.4 Envio das subconsultas 59
5.3.5 União dos resultados obtidos 60

5.3.6 Visualização dos resultados obtidos 60
5.4 Serviços oferecidos pelo protótipo mediador 60
5.5 Princípios para utilização do mediador proposto 60
5.6 Considerações Finais 63

6 DETALHAMENTO DO DESENVOLVIMENTO DO PROTÓTIPO DE MEDIADOR 64
6.1 Pacote analisador 64
6.2 Pacote gerenciaconexoes 67
6.3 Pacote gerenciaexecucao 68
6.4 Pacote gerencialinguagem 69
6.5 Pacote mediador 71
6.5.1 Processamento da consulta global no protótipo de mediador desenvolvido 72
6.6 Conversores 77

7 APLICAÇÃO PARA SIMULAÇÃO DE USO DO PROTÓTIPO DE MEDIADOR 79
7.1 Testes realizados 82
7.1.1 Execução de Consulta Global Simples 82
7.1.2 Execução de Consulta Global com Junções 84
7.1.3 Erro na Execução de Consulta Global 87

8 CONCLUSÃO 88

REFERÊNCIAS 90

ANEXO A ALGORITMO DE DECOMPOSIÇÃO DE CONSULTAS DO PROJETO *J-INTEGRATOR* 94

ANEXO B RELAÇÃO ENTRE O PADRÃO SQL-99 E A IMPLEMENTAÇÃO NOS SGBDS MYSQL E POSTGRESQL 96

ANEXO C DIAGRAMAS E-R DAS BASES DE DADOS PARA SIMULAÇÃO 98

ANEXO D ECG DAS BASES DE DADOS PARA SIMULAÇÃO . 100

LISTA DE ABREVIATURAS E SIGLAS

ANSI	*American Nacional Standards Institute* (Instituto Nacional Americano de Padrões)
BD	Banco de Dados
BDH	Banco de Dados Heterogêneo
CCG	Controlador de Concorrência Global
CCL	Controlador de Concorrência Local
DDL	*Data Definition Language* (Linguagem de Definição de Dados)
DML	*Data Manipulation Language* (Linguagem de Manipulação de Dados)
ECG	Esquema Conceitual Global
E-R	Entidade-Relacionamento
ISO	*International Standards Organization* (Organização Internacional de Padrões)
JDBC	Java *Database Connectivity*
LAN	*Local Area Network* (Rede Local)
LCG	Linguagem de Consulta Global
SGBD	Sistema Gerenciador de Banco de Dados
SGBDD	Sistema Gerenciador de Banco de Dados Distribuído
SGBDH	Sistema Gerenciador de Banco de Dados Heterogêneos
SQL	*Structured Query Language* (Linguagem de Consulta Estruturada)
TG	Transação Global
TL	Transação Local
XML	*Extensible Markup Language* (Linguagem Extensível de Marcação)

LISTA DE FIGURAS

Figura 1.1: Configuração básica de um Sistema de Banco de Dados. 14
Figura 1.2: Componentes básicos do processamento de consulta e administração de memória em um SGBD. 14
Figura 1.3: Passos do processamento de consultas. 15
Figura 1.4: Representação de um SGBD centralizado. 16
Figura 1.5: Representação de um SGBD distribuído. 16

Figura 2.1: Modelo de um sistema de banco de dados heterogêneo. 20
Figura 2.2: Modelo de um sistema de banco de dados federados. 21
Figura 2.3: Modelo de um sistema de armazém de dados. 22
Figura 2.4: Modelo de um sistema de mediação. 23
Figura 2.5: Processo *bottom-up* de integração de BDs. 28
Figura 2.6: Controladores de Concorrência Locais e Global. 31
Figura 2.7: Execução de uma transação global. 32
Figura 2.8: Arquitetura básica da integração baseada em linguagem de consulta. 34

Figura 3.1: Exemplo de arquitetura de mediação. 40
Figura 3.2: Arquitetura de mediação com a representação dos *wrappers*. . . . 42
Figura 3.3: Funcionamento da Camada de Mediação. 45

Figura 4.1: Funcionamento do processador de consultas em mediadores. . . . 52
Figura 4.2: Esquema da base de dados da revenda A. 52
Figura 4.3: Esquema da base de dados da revenda B. 53

Figura 5.1: Escopo da arquitetura de mediador e aplicação de simulação. . . . 55
Figura 5.2: Esquemas locais dos BDs hipotéticos para exemplificação do mapeamento XML. 61

Figura 6.1: Pacotes do protótipo de mediador desenvolvido. 64
Figura 6.2: Classes do protótipo de mediador desenvolvido. 65
Figura 6.3: Representação da estrutura para armazenamento das Conexões. . 68

Figura 6.4: Representação da estrutura para armazenamento dos resultados obtidos pelas unidades de execução. 69
Figura 6.5: Representação da estrutura para armazenamento do mapeamento do ECG. 70
Figura 6.6: Diagrama de Seqüência da execução da consulta global no protótipo de mediador. 75
Figura 6.7: Diagrama de Atividades da execução da consulta global no protótipo de mediador. 76
Figura 6.8: Diagrama de atividades da preparação do ambiente para a execução de consultas globais. 77

Figura 7.1: Janela principal do simulador após a execução de uma consulta global. 80
Figura 7.2: Janela de acompanhamento do log de execução. 80
Figura 7.3: Janela de acompanhamento da execução da subconsulta no PostgreSQL. 81
Figura 7.4: Janela de acompanhamento da execução da subconsulta no MySQL. 81

LISTA DE TABELAS

Tabela 4.1: Resumo e comparação das abordagens de Meng e Litwin. 50
Tabela 4.2: Consultas enviadas aos BDs das revendas A e B. 53

Tabela 5.1: Comandos aceitos pelo protótipo do mediador. 56
Tabela 5.2: Operadores lógicos e relacionais aceitos pelo protótipo do mediador. 56
Tabela 5.3: Demais operadores aceitos pelo protótipo do mediador. 57

Tabela 6.1: Exemplos de lexemas identificados em uma consulta definida conforme o padrão SQL-99. 66
Tabela 6.2: Elementos identificados na fragmentação da consulta global. . . . 71

Tabela 7.1: Resultados obtidos na execução da subconsulta enviada ao SGBD PostgreSQL - Consulta Global Simples 83
Tabela 7.2: Resultados obtidos na execução da subconsulta enviada ao SGBD MySQL - Consulta Global Simples 83
Tabela 7.3: Resultados obtidos na execução da consulta global após o processo de união - Consulta Global Simpes. 84
Tabela 7.4: Resultados obtidos na execução da subconsulta enviada ao SGBD PostgreSQL - Consulta Global com Junções. 85
Tabela 7.5: Resultados obtidos na execução da subconsulta enviada ao SGBD MySQL - Consulta Global com Junções. 86
Tabela 7.6: Resultados obtidos na execução da consulta global após o processo de união - Consulta Global com Junções. 86

RESUMO

Este trabalho apresenta uma proposta de arquitetura de um mediador para integração de bancos de dados heterogêneos com foco no processamento de consultas globais. Para embasamento dessa proposta é apresentado um estudo das possíveis formas de realizar essa integração e o detalhamento da arquitetura de mediadores e do processamento de consultas globais. Neste trabalho também é detalhada a implementação de um protótipo de mediador para bancos de dados heterogêneos e os resultados obtidos nos testes realizados com o protótipo gerado.

Palavras-chave: Banco de dados heterogêneo, Mediador, Processamento de consultas globais, SQL.

ABSTRACT

This paper presents a proposal of an heterogeneous database integration mediator architecture, focused on global queries processing. To base this proposal is presented a study of the possible ways to make this integration and a datailed explanation of the mediator architecture and the global queries prossessing as well. This paper also details the implementation of an heterogeneous database mediator prototype and the resuts obtained on tests of this prototype.

Keywords: Heterogeneous database, Mediator, Global query processing, SQL.

1 INTRODUÇÃO

De acordo com (LIMA, 2000), a quantidade de informação disponível tem crescido de forma exponencial. As informações armazenadas digitalmente são cada vez mais comuns sendo inúmeras as formas de armazená-las e manipulá-las. Segundo (LIMA; MELO, 1999), estas formas podem ser: repositórios de dados, arquivos convencionais, páginas da Web, bancos de dados, entre outras.

O desenvolvimento das tecnologias e das comunicações, em especial da Internet, tem afetado a forma das empresas fazerem negócios, o trabalho e até os relacionamentos interpessoais. Aliada a essa nova realidade surgiu a necessidade das informações de diversas fontes distintas serem acessadas por qualquer tipo de equipamento localizado em qualquer ponto do planeta (LIMA; MELO, 1999).

Uma das formas mais populares e eficientes de armazenamento de informações em meio digital é através de bancos de dados (SACRAMENTO et al., 2008). Os bancos de dados (BD) e, conseqüentemente, os sistemas gerenciadores de bancos de dados (SGBD) se tornaram ferramentas essenciais na realização de diversas atividades. Segundo (ELMASRI; NAVATHE, 2005), grande parte das pessoas se depara diariamente com alguma atividade que envolva interação com um banco de dados.

Um sistema gerenciador de bancos de dados é formado por um conjunto de dados e um conjunto de programas utilizados para acessar e manipular esses dados. O conjunto de dados é chamado de banco de dados e possui informações sobre uma determinada empresa ou assunto em particular (SILBERSCHATZ; KORTH; SUDARSHAN, 1999). Ainda (ELMASRI; NAVATHE, 2005) complementam essa definição enfatizando o fato de que os dados dessa coleção são relacionados entre si.

Segundo (SILBERSCHATZ; KORTH; SUDARSHAN, 1999), um SGBD é projetado para gerir grandes quantidade de informação. Esse gerenciamento de informações implica a definição de estruturas para armazenamento e mecanismos de acesso às informações armazenadas. O SGBD deve impedir tentativas de acesso não autorizado, prover o acesso simultâneo aos dados por diversos usuários e garantir a integridade das informações nos casos de falhas do sistema (SILBERSCHATZ; KORTH; SUDARSHAN, 1999).

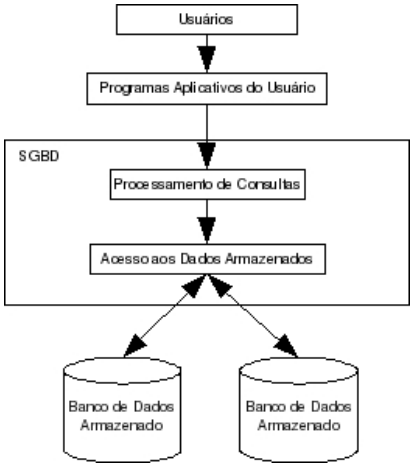

Figura 1.1: Configuração básica de um Sistema de Banco de Dados.

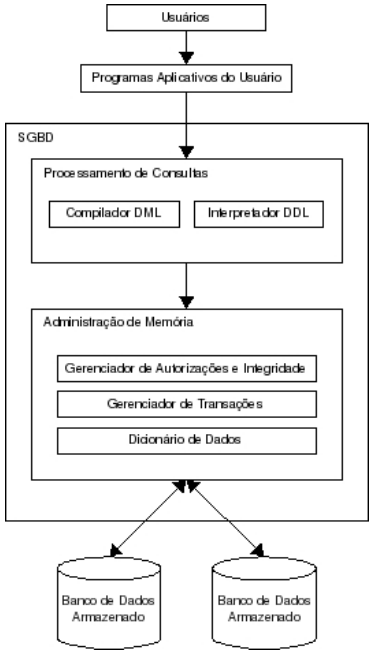

Figura 1.2: Componentes básicos do processamento de consulta e administração de memória em um SGBD.

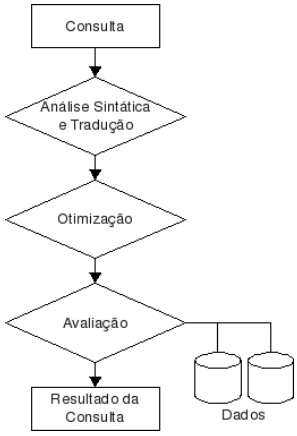

Figura 1.3: Passos do processamento de consultas.

A Figura 1.1 mostra, de forma simplificada, a configuração básica de um sistema de bancos de dados na visão de (ELMASRI; NAVATHE, 2005). Pode-se notar no esquema da Figura 1.1 que é possível separar de forma eficaz os dados da aplicação, facilitando, dessa forma, possíveis manutenções que o SGBD, o banco de dados ou a aplicação possam vir a sofrer.

Segundo (SILBERSCHATZ; KORTH; SUDARSHAN, 1999) os componentes que fazem parte de um SGBD podem ser divididos em componentes de processamento de consulta e componentes de administração de memória. Os principais componentes de cada categoria podem ser visualizados na Figura 1.2 (SILBERSCHATZ; KORTH; SUDARSHAN, 1999).

Os principais componentes de processamento de consulta são o compilador DML e o interpretador DDL. O compilador DML é responsável pela tradução dos comandos DML (Data Manipulation Language ou Linguagem de Manipulação de Dados) em instruções de baixo nível que serão executadas pelo processador enquanto o interpretador DDL é responsável pela interpretação de comandos DDL (Data Definition Language ou Linguagem de Definição de Dados) salvando os dados definidos no banco de dados (SILBERSCHATZ; KORTH; SUDARSHAN, 1999).

O processamento de consultas consiste na execução das atividades necessárias para realizar a extração dos dados armazenados em um BD (SILBERSCHATZ; KORTH; SUDARSHAN, 1999). Dentre essas atividades, são de maior importância a tradução da consulta do usuário, expressa em linguagem de alto nível, em expressões que podem ser processadas, a otimização, a tradução e a avaliação das consultas.

A figura 1.3 ilustra, de forma simplificada, o processo de execução de uma consulta de acordo com (SILBERSCHATZ; KORTH; SUDARSHAN, 1999). Nessa situação a consulta é verificada sintaticamente em busca de erros e, caso esteja correta, traduzida em uma linguagem baseada na álgebra relacional. Essa consulta traduzida é otimizada, ou seja, o melhor plano de avaliação é selecionado pelo SGBD para a

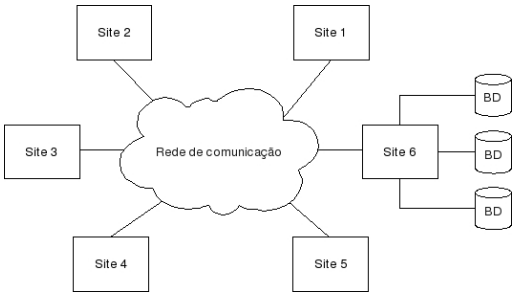

Figura 1.4: Representação de um SGBD centralizado.

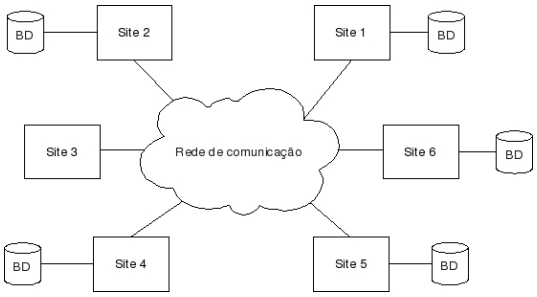

Figura 1.5: Representação de um SGBD distribuído.

execução da consulta. Escolhido o melhor plano de avaliação, esse é executado e o resultado é produzido e retornado para o usuário.

De acordo com (OLIVEIRA CUNHA, 2003), os SGBDs podem ser classificados de duas formas no que diz respeito à localização dos dados: centralizados ou distribuídos.

Em um BD centralizado todos os dados ficam localizados em um único local ou site. Entretanto define-se um banco de dados distribuído como "um conjunto de vários bancos de dados logicamente inter-relacionados, fisicamente separados em diferentes sites, dispersos geograficamente, distribuídos por uma rede de computadores" (OZSU; VALDURIEZ, 2001).

Segundo (OZSU; VALDURIEZ, 2001), (KORTH, 1995) e (CASANOVA, 1985), pode-se definir um SGBD distribuído (SGBDD) "como um sistema que possibilita o gerenciamento dos bancos de dados distribuídos e que torna a distribuição dos dados transparente para os usuários, como se fosse um sistema centralizado".

As Figuras 1.4 e 1.5 mostram as diferenças entre os SGBDs centralizados e distribuídos na visão de (OLIVEIRA CUNHA, 2003) e (OZSU; VALDURIEZ, 2001). A Figura 1.4 mostra diversos sites sendo que o SGBD está localizado somente em um deles. Já a Figura 1.5 mostra os diversos SGBDs localizados em diversos sites distintos. Em ambas as figuras os sites estão interligados através de uma rede.

Os SGBDDs podem ser homogêneos ou heterogêneos (OZSU; VALDURIEZ, 2001). No SGBD homogêneo, todos os SGBDs locais são semelhantes, ou seja,

não existem diferenças estruturais ou semânticas entre os BDs, independentemente do site onde se localizam. Já no caso dos SGBDs heterogêneos, podem existir dois ou mais SGBDs diferentes nos sites locais e são utilizados, de acordo com (CASANOVA, 1985), quando há necessidade de integrar sistemas já existentes.

De acordo com (ELMAGARMID; RUSINKIEWCZ; SHETH, 1999), os principais problemas presentes nos SGBDDs que podem interferir diretamente no processamento de consultas em SGBDs heterogêneos (SGBDH) são o formato dos dados e sua representação, o suporte a transações e as distintas linguagens de consultas implementadas pelos SGBDs.

O objetivo deste trabalho é idealizar uma proposta de integração de bancos de dados heterogêneos através de um mediador genérico cuja arquitetura também será foco de estudo, proposição e implementação. Este mediador proposto estará focado no tratamento de uma linguagem de consulta global, uma das principais etapas desse tipo de integração de fontes de informação heterogêneas. Essa proposta de arquitetura considerará a utilização de SBGDs relacionais, que estão baseados na linguagem SQL padrão para a realização de consultas. A estrutura de integração utilizada valer-se-á do uso de esquemas conceituais globais já existentes para a realização das consultas e não terá foco no gerenciamento de transações, na implementação de dispositivos de segurança dos dados ou em técnicas avançadas de junção e processamento dos dados obtidos.

Este trabalho tem como principal motivação a dificuldade de se encontrar um mediador na linguagem Java, que possa ser utilizado no desenvolvimento de aplicações. Esse mediador terá como objetivo fornecer uma camada de acesso transparente aos diversos BDs que a aplicação necessite acessar.

A divisão deste trabalho deu-se de forma a especializar a informação, partindo das diversas formas de integração de BDHs até chegar ao funcionamento de um mediador. No capítulo 02 são apresentados os conceitos e as formas de realizar a integração de BDHs, como forma de embasamento para o restante do estudo e proposição da arquitetura do protótipo de mediador a ser desenvolvido.

No capítulo 03 são apresentados os conceitos envolvidos no processo de integração de BDHs utilizando-se mediadores e o funcionamento dessa arquitetura. Além disso, são apresentados dois projetos que utilizam essa abordagem para integrar BDHs, mostrando que a abordagem já está consolidada e sendo utilizada em alguns projetos.

Ao capítulo 04 cabe o detalhamento do processamento de consultas com a utilização de mediadores. Este capítulo também é responsável por mostrar duas das possíveis abordagens utilizadas para o processamento de consultas, principal foco do estudo desse trabalho.

No capítulo 05 é realizada a proposição e o detalhamento da arquitetura de um mediador com foco no processamento das consultas globais. Neste capítulo também

é descrito o escopo do desenvolvimento do protótipo desse mediador, detalhando as estruturas e os testes realizados utilizando-se esse protótipo.

O detalhamento do desenvolvimento do protótipo do mediador, baseado na arquitetura definida no capítulo 05, é apresentada no capítulo 06. Juntamente com o protótipo do mediador houve o desenvolvimento de uma aplicação de simulação que utiliza o mesmo. As principais funcionalidades desse simulador são apresentadas no capítulo 07.

Finalmente, no oitavo capítulo é realizada a conclusão dos estudos realizados, dando uma perspectiva aos estudos e trabalhos futuros que podem ser originados a partir deste e da bibliografia da área.

2 SISTEMAS DE BANCOS DE DADOS HETEROGÊNEOS

De acordo com (BOUGUETTAYA; BENATALLAH; ELMAGARMID, 1998), Sistemas Gerenciadores de Bancos de Dados Heterogêneos (SGBDH) são sistemas de informação que fornecem diversos graus de integração entre vários BDs. Esse SGBDH funciona como o gerenciador de vários SGBDs que são chamados também de SGBDs locais ou componentes. Cada um desses SGBDs locais possui o seu BD local e sua própria plataforma. Dessa forma, a união de todos os esses BDs irá formar o que é chamado de Banco de Dados Heterogêneos (BDH) (ARAUJO DUARTE, 2004). De forma mais direta, um BDH é o resultado da integração e união de informações oriundas de diferentes BDs que são gerenciados por diferentes SGBDs.

Como complemento dos conceitos do parágrafo anterior pode-se utilizar algumas idéias de (GARCIA-MOLINA; ULMAN; WIDOM, 2001) que colocam, do ponto de vista de uma integração mais genérica, que os aplicativos de integração de dados recebem dados de diversas origens e constroem, a partir dessas origens, um grande banco de dados, ou seja, um BDH. Esse BDH geralmente é virtual, ou seja, essas informações continuam em suas fontes e não são duplicadas para outra fonte. Dessa forma os dados podem ser consultados como se estivessem em uma única fonte de informação (GARCIA-MOLINA; ULMAN; WIDOM, 2001). A Figura 2.1 apresenta um exemplo de composição de um SGBDH (ARAUJO DUARTE, 2004).

Segundo (ARAUJO DUARTE, 2004), a heterogeneidade pode ocorrer em qualquer nível do BD, envolvendo diferentes elementos que vão desde a implementação do SGBD até o conteúdo do BD. Por exemplo, SGBDs diferentes podem utilizar linguagens de consulta e manipulação de dados distintas, modelos de dados e plataformas diferentes.

De acordo com (ARAUJO DUARTE, 2004), os BDHs constituem a integração de vários BD pré-existentes. Geralmente o BDH é projetado de baixo para cima (Figura 2.1), partindo-se dos BDs locais existentes e criando-se estruturas de integração sobre eles, em um ou mais níveis (estrutura *bottom-up*). A integração das informações deve ocorrer sem que haja qualquer modificação nos dados do BDs locais existentes. É

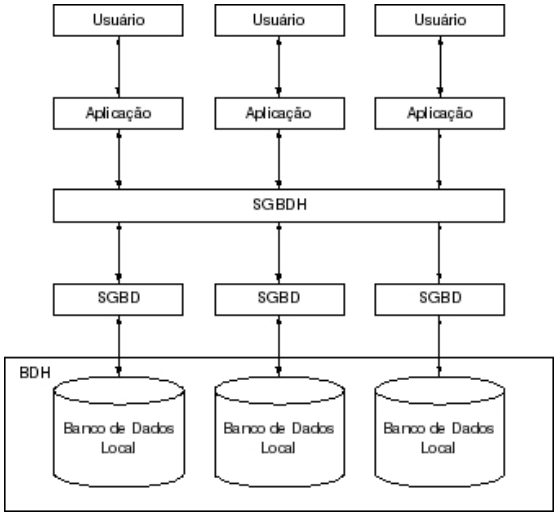

Figura 2.1: Modelo de um sistema de banco de dados heterogêneo.

importante salientar que as informações dos BDs locais podem ser redundantes e inconsistentes, logo, cabe ao SGBDH unir essas informações, apresentando soluções para esses problemas (ARAUJO DUARTE, 2004).

A principal e mais importante característica de um SGBDH, destacada por (ARAUJO DUARTE, 2004), é o fato dos BDs locais manterem sua autonomia, ou seja, eles continuam mantendo total controle sobre os dados e processamento locais. Do ponto de vista do usuário, (ARAUJO DUARTE, 2004) friza o fato dos usuários de SGBDHs poderem acessar um, vários ou todos os BDs locais de forma transparente em uma única consulta.

Em SGBDHs a integração de novos sistemas e plataformas de hardware e software devem ser possíveis sem que seja necessário reestruturar as implementações já existentes (DARONCO, 2000). Esta característica é um dos principais objetivos dos SGBDHs.

2.1 Métodos de Integração de Bancos de Dados

De acordo com (GARCIA-MOLINA; ULMAN; WIDOM, 2001), bancos de dados federados, armazéns de dados e mediação são as formas mais utilizadas para realizar a integração de BDs.

2.1.1 Bancos de Dados Federados

Neste método de integração, as origens de dados são independentes, entretanto, uma origem pode solicitar que outra lhe forneça informações (GARCIA-MOLINA; ULMAN; WIDOM, 2001). Essa é a forma mais simples de integração de BDs pois é implementada através de conexões de um BD para outro entre todos os pares de

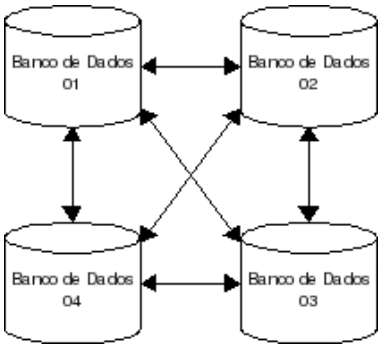

Figura 2.2: Modelo de um sistema de banco de dados federados.

bancos de dados que necessitam trocar informações. Essas conexões permitem que um BD1 consulte outro BD2, de forma que BD2 entenda o que BD1 deseja consultar.

O maior problema desse método de integração é a quantidade de fragmentos de código que necessitam ser escritos para prover a comunicação entre os diversos BDs (GARCIA-MOLINA; ULMAN; WIDOM, 2001). Por exemplo, se n BDs necessitam se comunicar com cada um dos *n-1* outros BDs, serão necessários *n(n-1)* fragmentos de código para suportar essa comunicação.

A figura 2.2 (GARCIA-MOLINA; ULMAN; WIDOM, 2001) representa essa situação com quatro BDs que necessitam se comunicar. Nessa situação são necessários doze fragmentos de código para coverter consultas de um BD para outro, tendo em vista que cada um dos SGBDs envolvidos pode trabalhar com um dialeto SQL próprio e cada BD tem seu próprio esquema. Isso faz com que, segundo (GARCIA-MOLINA; ULMAN; WIDOM, 2001), para cada consulta necessária, seja gerado uma grande quantidade de código para que os BDs possam trocar informações.

2.1.2 Armazéns de Dados

Nesta situação de integração, segundo (GARCIA-MOLINA; ULMAN; WIDOM, 2001), os dados dos diversos BDs são copiadas para um único BDH, chamado de *armazém*, também conhecido como *data warehouse*. Este método, de acordo com (GARCIA-MOLINA; ULMAN; WIDOM, 2001), pode exigir que os dados sejam, de alguma forma, transformados para obedecerem ao esquema do armazém. Além disso os armazéns de dados necessitam ser constantemente atualizados, caso contrário, poderão conter informações incorretas ou desatualizadas.

Uma vez que os dados estejam no armazém, as consultas aos mesmos podem ser emitidas para o armazém. Entretanto, de acordo com (GARCIA-MOLINA; ULMAN; WIDOM, 2001), os usuários não possuem permissão para realizar alterações nos dados. Isso ocorre devido à natureza dos armazéns de dados, que são construídos essencialmente para consultas.

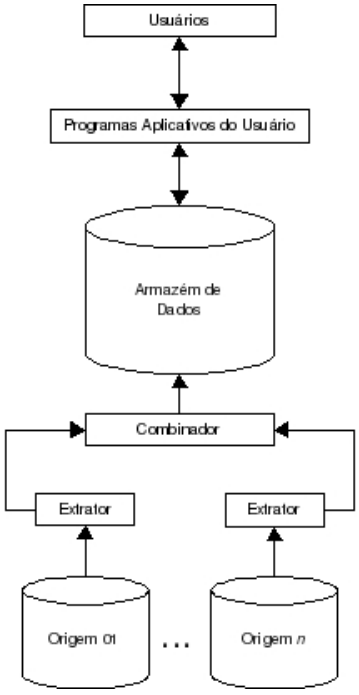

Figura 2.3: Modelo de um sistema de armazém de dados.

Existem três abordagens mais usuais para construir armazéns de dados, segundo (GARCIA-MOLINA; ULMAN; WIDOM, 2001).

- A abordagem mais comum é aquela em que o armazém é constantemente reconstruído a partir dos dados das origens. Geralmente essa reconstrução acontece em períodos em que haja pouco acesso ao armazém como, por exemplo, aos finais de semana. Essa abordagem apresenta como desvantagens o tempo de reconstrução do armazém e a possibilidade dos dados do mesmo tornarem-se desatualizados antes da reconstrução do armazém.

- Outra possível abordagem é aquela na qual o armazém é atualizado periodicamente. Por exemplo, essa reconstrução pode ocorrer toda a noite. As atualizações do armazém consideram somente as útimas mudanças nos dados. Nessa situação o volume de dados envolvidos é menor, facilitanto a atualização e reduzindo o tempo dessa. A principal desvantagem dessa abordagem está relacionada à quais dados deverão ser atualizados, pois esse cálculo possui um custo computacional muito elevado.

- Como alternativa para a construção de armazéns de dados, pode-se alterar esses a cada mudança nas origens de dados. Essa abordagem exige muita comunicação e processamento. Essa é uma abordagem que continua sendo fruto de muitos estudos.

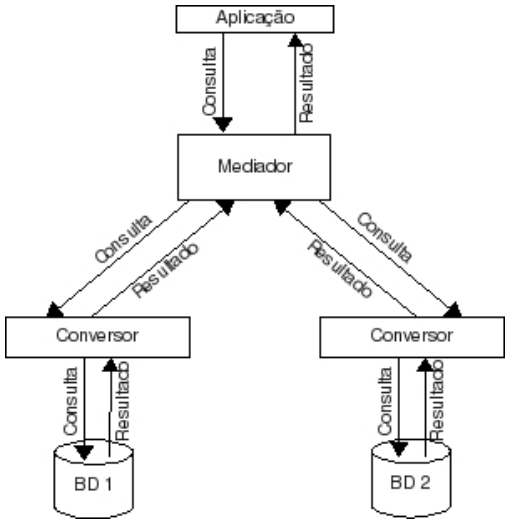

Figura 2.4: Modelo de um sistema de mediação.

2.1.3 Mediação

"Um mediador é um componente de software que admite um *banco de dados virtual*, que o usário pode consultar como se ele estivesse *materializado*, ou seja, fisicamente construído." (GARCIA-MOLINA; ULMAN; WIDOM, 2001). Este conceito resume, de forma objetiva, o funcionamento de um mediador. O mediador basicamente funciona como uma camada de acesso às informações de um BD. Essa camada não armazena dados, somente controla o acesso a esses através de informações sobre os dados (metadados). Quando a aplicação consulta alguma informação dessa camada, essa converte a consulta em uma ou mais consultas e as envia às diversas origens de dados. O mediador então realiza o processamento e junção dos dados obtidos das diversas fontes e devolve ao usuário um único conjunto de informações (GARCIA-MOLINA; ULMAN; WIDOM, 2001). O esquema da figura 2.4 exemplifica o funcionamento da arquitetura de mediação segundo (GARCIA-MOLINA; ULMAN; WIDOM, 2001).

2.2 Problemas de Integração de Informações

O processo de integração de informações de distintas fontes apresenta alguns problemas que devem ser considerados. Segundo (GARCIA-MOLINA; ULMAN; WIDOM, 2001), esses problemas devem ser considerados independentemente do método de integração que será implementado, pois exercem alguma influência na execução de consultas sobre os dados armazenados. De acordo com (ARAUJO DUARTE, 2004), (BALLARDIN, 2004) e (GARCIA-MOLINA; ULMAN; WIDOM, 2001), os problemas a serem considerados na integração de informações são distribuição de dados, autonomia dos SGBDs, interoperabilidade, diferenças nos tipos de dados, nos

valores, na semântica e omissão de valores.

2.2.1 Distribuição dos Dados

Nos dias atuais, de acordo com (ARAUJO DUARTE, 2004), a distribuição dos dados está cada vez mais presente nas aplicações. Isso se deve ao fato da maior parte dos computadores estarem conectados a uma rede (LAN ou Internet). Neste caso é comum a necessidade de combinar fontes de dados localizadas fisicamente em diferentes servidores. Essa situação acaba sendo um dificultador no momento de integrar informações de fontes de dados distintas (BALLARDIN, 2004).

As possíveis classificações para a distribuição dos dados são, de acordo com (ARAUJO DUARTE, 2004):

- Dados Duplicados: Ocorrem quando cópias do mesmo dado são mantidas em fontes diferentes;
- Subconjuntos de Dados: Ocorrem quando os dados armazenados em estações de trabalho são parte dos dados armazenados nos servidores;
- Dados Reorganizados: Ocorrem quando os dados de produção são derivados de sistemas de suporte à decisão;
- Dados Particionados: Ocorrem quando uma mesma estrutura é utilizada em diferentes locais, porém armazenando dados distintos. Por exemplo, nos caso de BDs distribuídos, podem haver tabelas fracionadas, ou seja, parte dos dados de uma tabela estão em um servidor e parte estão em outro (ABREU, 2006);
- Dados de Esquemas Separados: Ocorrem quando estruturas de dados distintas são utilizadas em locais diferentes;
- Dados Incompatíveis: Ocorrem quando os dados são acessados pelos usuários por meio de sistemas distintos.

A distribuição de dados, segundo (DATE, 2004), utiliza elevado tráfego de rede, causando muito *overhead* na obtenção de informações. Com o objetivo de minimizar esse tráfego, é necessário realizar um processo de otimização das consultas. Essa otimização ocorre de forma distribuída em SGBD homogêneos (DATE, 2004), ocorrendo primeiramente uma otimização global das consultas e após as otimizações locais em cada SGBD local envolvido. Entretanto, no caso dos SGBDH, devido aos diferentes SGBDs envolvidos e a autonomia que esses possuem, o processamento e a otimização necessária às consultas tornam-se demasiadamente complexas (SILBERSCHATZ; KORTH; SUDARSHAN, 1999). Nessa situação torna-se inviável a otimização das consultas em nível global, utilizando-se usualmente somente a otimização a nível local.

2.2.2 Autonomia dos SGBDs

A autonomia dos SGBDs indica até que ponto eles podem operar individualmente (BALLARDIN, 2004). Essa autonomia pode relacionar-se a fatores como troca ou não de informações pelos sistemas componentes, execução de transações de forma independente, entre outros.

É possível ainda, de acordo com (BALLARDIN, 2004), possuir-se autonomia de projeto, de comunicação, de execução e de associação, influenciando na execução das consultas aos dados do BDH. A autonomia de projeto refere-se ao fato de um SGBD possuir seu próprio modelo de dados, sua linguagem de consulta, suas restrições, entre outras características. A autonomia de comunicação significa que cada SGBD gerencia como e quando vai responder às solicitações dos demais SGBDs (quando for o caso). Já a autonomia de execução pode ser percebida quando cada SGBD pode decidir sobre a ordem de execução das transações. A autonomia de associação é vista quando um SGBD pode gerenciar quais dados serão compartilhados com quais usuários.

2.2.3 Interoperabilidade

"A interoperabilidade entre SGBDs está relacionada à habilidade de solicitar e receber serviços entre os sistemas relacionados" (BALLARDIN, 2004). Um sistema é interoperável se consegue trocar mensagens, receber serviços e operar como uma unidade na solução de um problema comum (BALLARDIN, 2004).

O processamento das consultas em um SGBDH é afetado diretamente pela interoperabilidade entre os SGBDs componentes (OZSU; VALDURIEZ, 2001). Pode ocorrer tratamento não uniforme das consultas nos SGBDs componentes devido às diferentes capacidades desses. Além disso, a interoperabilidade está intimamente relacionada aos mesmos fatores de autonomia dos SGBDs que influenciam no processamento de consultas (OZSU; VALDURIEZ, 2001).

2.2.4 Diferenças nos tipos de dados

De acordo com (GARCIA-MOLINA; ULMAN; WIDOM, 2001), ocorre quando em BDs diferentes, um mesmo dado é representado por tipos de dados diferentes. Por exemplo, pode-se citar um cadastro de equipamentos eletrônicos em que em um BD1 o número de série seja representado por uma seqüência de caracteres, enquanto em um BD2 seja representado por um número inteiro.

2.2.5 Diferenças nos valores

É colocado por (GARCIA-MOLINA; ULMAN; WIDOM, 2001) que este problema ocorre quando um mesmo conceito é representado por diferentes valores em BDs distintos. Por exemplo, pode-se citar um cadastro de veículos onde a cor de um

determinado veículo seja representado pela seqüência de caracteres *VERMELHO* e em outro pelo valor inteiro *1* e ainda um terceiro BD onde seja representado pelos caracteres *VER*. O exemplo dessa situação ainda pode ser agravado se houver uma quarta fonte onde os caracteres *VER* representem a cor verde.

2.2.6 Diferencas na semântica

De acordo com (GARCIA-MOLINA; ULMAN; WIDOM, 2001), diferenças de semântica ocorrem quando termos importantes do domínio do BD recebem iterpretações diferentes. Por exemplo, no caso de um cadastro de veículos, um BD poderia incluir carros e caminhões na tabela de veículos enquanto um segundo BD teria tabelas separadas para carros e caminhões.

2.2.7 Omissão de valores

Ocorre quando um BD deixa de registrar informações enquanto os demais BDs registram a mesma informação (GARCIA-MOLINA; ULMAN; WIDOM, 2001). Por exemplo, um determinado BD de veículos poderia registrar as cores dos mesmos enquando um segundo BD não o faria.

Esses últimos problemas (diferenças nos tipos de dados, valores e semântica e omissão de valores) não influenciam diretamente a execução da consulta, mas sim o conjunto de dados obtido por essa (GARCIA-MOLINA; ULMAN; WIDOM, 2001).

2.3 Abordagens para Integração de Bancos de Dados Heterogêneos

De acordo com (ARAUJO DUARTE, 2004), a integração de diversos SGBDs locais pode ser realizada envolvendo diferentes elementos do SGBD. Neste tópico serão apresentadas as três abordagens mais usuais para integração de BDHs.

2.3.1 Integração Baseada em Esquemas

Segundo (ELMASRI; NAVATHE, 2005), o conceito de esquema em BDs foi incorporado a partir da SQL2[1] e possui a função de ser um agrupador das tabelas e outros objetos que pertençam à mesma aplicação de um BD. Os esquemas em geral são identificados por um nome e possuem uma indentificação de autorização, que indica usuário proprietário do esquema e os descritores de todos os elementos que pertencem ao esquema (ELMASRI; NAVATHE, 2005).

[1]De acordo com (ELMASRI; NAVATHE, 2005), a linguagem SQL teve seu primeiro padrão publicado graças a um esforço da ANSI e da ISO, em 1986, o qual ficou conhecido como SQL-86 ou SQL1. Uma versão revisada e expandida foi desenvolvida em 1992, ficando conhecida por SQL-92 ou SQL2. Por fim, em 1999 a terceira versão da linguagem foi lançada, a SQL3 ou SQL-99, como é popularmente conhecida.

De acordo com (ARAUJO DUARTE, 2004), o maior desafio para realizar a integração de esquemas é interpretar as informações remotas, ou seja, as que se encontram nos BDs componentes, e traduzi-las para um contexto local. Isso leva a alguns conflitos existentes nos BDHs, dos quais destacam-se, segundo (ARAUJO DUARTE, 2004):

- *Conflito de Nomes*: ocorre quando tipos de dados idênticos possuem nomes diferentes ou então quando tipos de dados diferentes possuem os mesmos nomes;
- *Diferentes Modelos de Dados*: ocorre quando uma mesma informação é representada de formas distintas nos BDs componentes, por exemplo, no caso de um endereço que pode ser representado por uma tabela em um esquema e como um atributo de uma tabela em outro;
- *Conflito de chaves e índices*;
- *Incompatibilidade de tipos de dados*;
- *Diferentes tipos de restrições para os dados.*

São várias as técnicas que podem ser utilizadas para integrar informações por meio da integração de esquemas, entretanto, todas elas deverão passar pelas etapas a seguir para realizar a integração (BATINI; LENZERINI; NAVATHE, 1986) (ARAUJO DUARTE, 2004).

- *Pré-integração*: Nesta fase os esquemas são analisados. Com base nessa análise são definidas as regras e políticas de integração, quais e quantos são os esquemas a serem integrados, qual a ordem de integração e as restrições que serão utilizadas. Cada componente do BD é traduzido para um modelo de dados comum, também chamado de modelo canônico, são especificadas as restrições globais identificados conflitos de dados (sinônimos e homônimos);
- *Comparação de esquemas*: Os esquemas são comparados para determinar correspondências e possíveis conflitos. Nesse passo todos os atributos, entidades e relações entre os tipos são comparados para verificar se são idênticos, equivalentes, compatíveis, incompatíveis ou relacionados de alguma forma;
- *Correspondência*: Após a identificação dos conflitos, é necessário resolvê-los de alguma forma para tornar os esquemas compatíveis e integráveis. Nessa fase é necessária grande participação dos desenvolvedores e/ou usuários dos BDH, uma vez que não há grande êxito na utilização de ferramentas automáticas de resolução de conflitos;

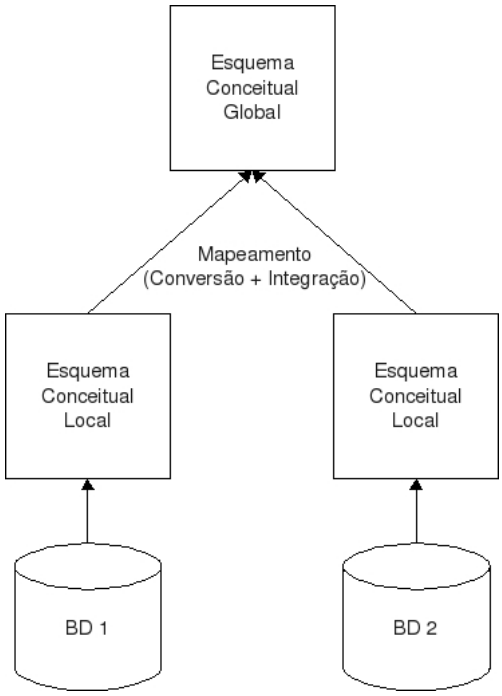

Figura 2.5: Processo *bottom-up* de integração de BDs.

- *União e reestruturação*: É a última fase do processo de transformação dos esquemas. Os esquemas componentes são unidos em um esquema integrado ou **esquema conceitual global** (ECG), que engloba todas as estruturas para acessar os esquemas componentes. Realizada a construção do ECG, esse pode ser testado considerando-se as seguintes propriedades qualitativas: completude e corretude, minimalidade e inteligibilidade. Essas propriedades serão explicadas ao longo dessa subseção.

De acordo com (OZSU; VALDURIEZ, 2001), o ECG dos BDH apenas representa uma coleção de alguns BDs locais que cada SGBD local deseja compartilhar. A definição de um ECG é importante para existir um único tipo de esquema para consulta ao BDH (ARAUJO DUARTE, 2004). Dessa forma, o que ocorre é um mapeamento dos esquemas locais para um esquema global.

É sabido, de acordo com (OZSU; VALDURIEZ, 2001), que o projeto de SGBDH é *bottom-up*, ou seja, parte-se dos bancos de dados já existentes até obter-se o BDH. Com relação à integração desses bancos de dados já existentes, (OZSU; VALDURIEZ, 2001) coloca que essa pode ser realizada em duas fases, que compreendem as etapas anteriormente explicitadas: conversão e integração de esquemas. A Figura 2.5 (OZSU; VALDURIEZ, 2001) exibe o processo de integração de esquemas, exemplificando o processo *bottom-up*.

2.3.1.1 Conversão de Esquemas

Segundo (OZSU; VALDURIEZ, 2001), a conversão de esquemas consiste no mapeamento de um esquema para outro (esquema local para global), especificando-se um modelo de dados destino para o ECG (representação canônica). As combinação das etapas de conversão e integração fornece ao integrador todas as informações sobre o BD global de uma única vez.

Um ponto que merece destaque na conversão de esquemas, na visão de (OZSU; VALDURIEZ, 2001), é o estabelecimento de equivalências entre os conceitos dos esquemas de origem e os conceitos do esquema destino.

A realização da conversão de esquemas deve considerar duas situações dificultadoras, de acordo com (OZSU; VALDURIEZ, 2001). A primeira dificuldade é a determinação de quais relações representam entidades e quais representam relacionamentos. Essas informações costumam ser de fácil identificação se houver relações específicas que representam relacionamentos ou entidades. Caso não seja esse o caso, pode-se identificá-las a partir das chaves estrangeiras definidas para as relações. Quando essa determinação é realizada, o mapeamento ocorre de forma direta, ou seja, as relações que representam entidades são mapeadas como entidades e as que representam relacionamentos são mapeadas como relacionamentos. A segunda dificuldade relaciona-se à natureza dos relacionamentos, uma vez que pode-se ter uma tabela em um BD com um relacionamento um-para-muitos e uma tabela equivalente em outro BD com um relacionamento muitos-para-muitos, por exemplo (ARAUJO DUARTE, 2004).

2.3.1.2 Integração de Esquemas

É nesta fase que o ECG é propriamente gerado, segundo (ARAUJO DUARTE, 2004) e (OZSU; VALDURIEZ, 2001), integrando os esquemas locais. A integração de esquemas consiste em identificar os componentes de um BD relacionados a outros, selecionar a melhor representação para o ECG e integrar os componentes de cada esquema local.

A integração de esquemas, de acordo com os estudos de (OZSU; VALDURIEZ, 2001), envolve duas tarefas, a *homogeneização* e a *integração*. Na homogeneização são detectados problemas estruturais e semânticos de cada um dos BDs componentes. O objetivo principal dessa homogeneização é certificar que os BDs locais são comparáveis entre si, tanto na estrutura quanto na semântica. A tarefa de integração ocorre após a homogeneização e é responsável pela mesclagem dos esquemas dos vários BDs locais, criando finalmente o ECG.

Homogeneização: De acordo com (OZSU; VALDURIEZ, 2001), nesta fase são resolvidos os problemas de heterogeneidade semântica e estrutural. Entende-se por

heterogeneidade semântica as diferenças que dizem respeito ao significado, à interpretação e à aplicação dos dados, apresentando-se geralmente como conflitos de nomenclatura. O principal conflito de nomenclatura é o de *sinônimos* e *homônimos*.

Conflitos estruturais podem ocorrer de quatro formas segundo (OZSU; VALDURIEZ, 2001): *conflitos de tipo*, *conflitos de dependência*, *conflitos de chave* e *conflitos comportamentais*.

Como é explicitado por (OZSU; VALDURIEZ, 2001), a homegeneização necessita de um alto grau de interação humana, pois se faz necessário o conhecimento semântico sobre todos os esquemas envolvidos no processo. Por exemplo, é necessário conhecer os significados das informações armazenadas para saber se dois atributos são idênticos.

Integração: Nesta fase, na visão de (OZSU; VALDURIEZ, 2001), é realizada a mesclagem dos esquemas intermediários e sua reestruturação. Todos os esquemas são mesclados em um único esquema e após reestruturados, criando o melhor esquema integrado, o ECG. Esse processo de mesclagem exige que todas as informações contidas nos esquemas locais sejam preservadas no ECG.

2.3.2 Integração Baseada em Transações

Em SGBDs, muitos usuários necessitam ler e atualizar informações concorrentemente, ou seja, ao mesmo tempo (ARAUJO DUARTE, 2004). Cabe ao controle de concorrência gerenciar todas essas operações de forma que elas ocorram de forma correta, levando o BD sempre a um estado consistente, ou seja, correto.

Uma transação é conceituada por (SILBERSCHATZ; KORTH; SUDARSHAN, 1999) como "uma unidade de execução de programa que acessa e, possivelmente, atualiza vários itens de dados". Como complemento desse conceito, pode-se considerar a idéia de (DATE, 2004), que enfatiza o fato das transações serem formadas por diversas operações de leitura e escrita de dados, levando o BD de um estado correto a outro estado correto sem garantir a correção dos dados nos pontos intermediários, ou seja, durante a execução da transação.

Existem algumas propriedades das transações que são exigidas na implementação de SGBDs. Este trabalho somente apresentará o conceito dessas propriedades por não ser o foco do estudo a que se propõe. Essas propriedades são responsáveis por assegurar a integridade dos dados (SILBERSCHATZ; KORTH; SUDARSHAN, 1999). Essas propriedades são chamadas de propriedades ACID em virtude da primeira letra do nome de cada uma delas:

- *Atomicidade*: Essa propriedade garante que ou são realizadas todas as operações de uma transação ou nenhuma será;
- *Consistência*: A execução de uma transação preserva a consistência do con-

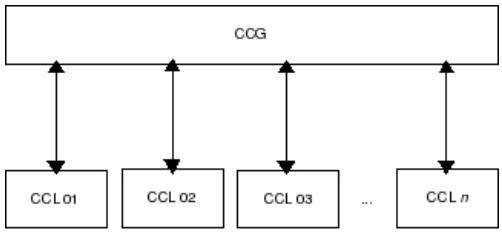

Figura 2.6: Controladores de Concorrência Locais e Global.

junto de dados;

- *Isolamento*: Diversas transações podem ser executadas de forma concorrente. Essa propriedade garante que uma transação em execução não saberá que existem outras transações concorrentes no sistema. Dessa forma as transações são isoladas umas das outras e executam como se fossem únicas;

- *Durabilidade*: Garante que, após a execução com sucesso de uma transação, as mudanças que essa efetuou sobre os dados persistem, mesmo que haja falhas no sistema.

O estabelecimento de estratégias de controle de concorrência em SGBDHs é mais complexo que em SGBDs homogêneos. Isso ocorre devido à preocupação com a heterogeneidade e autonomia dos SGBDs, além da distribuição dos dados. Por exemplo, em SGBDHs os controladores de concorrência locais (CCL) são desenvolvidos de forma que garantam suas independências dos demais SGBDs locais. Outra situação que dificulta o controle de transações em SGBDHs é que o controlador de concorrência global (CCG) necessita de informações da execução das transações a nível local. Entretanto, o CCG não possui acesso a essas informações e não pode forçar os CCLs a fornecê-las. Existe ainda uma terceira situação colocada por (ARAUJO DUARTE, 2004) que interfere nesse controle de concorrência. Esse terceiro fato diz respeito ao fato dos CCLs tomarem as decisões sobre as operações de *commit* e *rollback*[2] considerando somente seus respectivos estados. A figura 2.6 mostra os controladores de concorrência em seus dois níveis, na visão de (OZSU; VALDURIEZ, 2001): local e global.

Em um SGBDH, de acordo com (OZSU; VALDURIEZ, 2001), existem dois níveis de transação: transações locais (TL), que são submetidas a cada SGBD local e

[2]De acordo com (SILBERSCHATZ; KORTH; SUDARSHAN, 1999), as linguagens de manipulação de dados devem possuir construtores para especificar quais comandos são parte de uma transação. O padrão SQL-92 especifica dois possíveis comandos para encerrar uma transação: **commit work** e **rollback work**. A palavra-chave *work* é opcional em ambos os comandos. O primeiro comando efetiva a transação corrente e inicia uma nova. O segundo, aborta a transação corrente. Caso uma transação seja encerrada sem um desses comandos, as operações realizadas pela transação são efetivadas ou desfeitas, dependendo da implementação do SGBD.

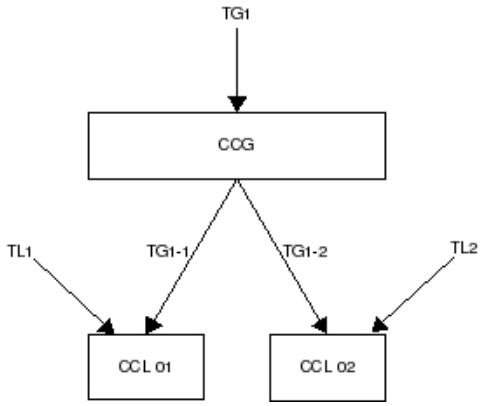

Figura 2.7: Execução de uma transação global.

as transações globais (TG), que são submetidas ao SGBDH. As transações locais manipulam dados de um único BD enquanto as globais o fazem com dados de vários BDs, ou seja, do BDH. A execução de uma transação em um SGBDH, de acordo com (OZSU; VALDURIEZ, 2001), dá-se da seguinte forma: a transação global é dividida em um conjunto de subtransações. Cada uma dessas subtransações, ao ser enviada para um SGBD local, passa a ser uma transação local desse SGBD.

A figura 2.7 esquematiza a execução de uma transação global, na visão de (OZSU; VALDURIEZ, 2001). É necessário observar que a transação global TG1 foi dividida nas subtransações TG1-1 e TG1-2. Além disso, podem existir outras transações enviadas diretamente aos SBGDs locais. Essas transações são representadas por TL1 e TL2.

A execução de transações globais requer que os algoritmos de controle de concorrência garantam as propriedades ACID (OZSU; VALDURIEZ, 2001). Entretanto, devido à autonomia dos SGBDs locais, não é tarefa fácil manter essas propriedades. Os algoritmos de controle de concorrência realizam a sincronização das transações concorrentes de modo a ordenar as operações conflitantes mantendo uma ordem de serialização[3] entre as transações.

Conforme colocado por (OZSU; VALDURIEZ, 2001), a execução concorrente de duas ou mais transações pode entrar em conflito se realizam operações sobre os mesmos dados e uma dessas operações é uma gravação. Os conflitos podem ser de leitura-gravação, quando uma transação tentar ler um dado enquanto outra realiza uma gravação sobre o mesmo dado, ou então de gravação-gravação, quando duas transações tentam realizar gravações sobre os mesmos dados. Nestes casos, ocorre que duas TGs tratadas pelo CCG podem não parecer conflitantes. Entretanto, a existência de transações locais pode implicar conflitos de transações nos BDs locais.

[3]Segundo (DATE, 2004), a execução de duas ou várias transações é *serializável* se e somente se ocorrer corretamente e a execução das transações for equivalente à execução das mesmas de forma serial, ou seja, uma de cada vez, em uma determinada seqüência, obtendo-se os mesmos resultados.

Esses conflitos, chamados de indiretos por (OZSU; VALDURIEZ, 2001), não podem ser detectados pelo CCG e são um dificultador na implementação de um SGBDD, seja esse homogêneo ou heterogêneo.

De acordo com (OZSU; VALDURIEZ, 2001), foram definidas condições para transações globais poderem atualizar com segurança dados em um SGBDD. A primeira condição é exigir que os SGBDs locais garantam a atomicidade de sincronização local, ou seja, os CCL são responsáveis por executar corretamente as transações em seus BDs locais. Além disso, cada SGBD local é responsável por manter seu escalonamento serializável e recuperável. A segunda condição exige que cada CCL mantenha a ordem de execução relativa das transações determinadas pelo CCG. Dessa forma o CCG fica responsável por coordenar a transmissão das subtransações globais aos CCL e por coordenar a sua execução. O CCG ainda fica responsável por lidar com impasses globais que ocorrem entre transações globais.

2.3.3 Integração Baseada em Linguagem de Consulta

Na visão de (ARAUJO DUARTE, 2004), é possível utilizar soluções baseadas na utilização de uma linguagem global de acesso aos dados para realizar a integração de BDHs. Nessa abordagem de integração existe uma Linguagem de Consulta Global (LCG) e um *Processador de Consultas*, responsável por traduzir e quebrar a consulta global em subconsultas que são enviadas aos SGBDs locais.

Cabe ainda ao processador de consultas, de acordo com (ARAUJO DUARTE, 2004), realizar a junção dos dados obtidos dos BDs locais, devolvendo à camada de aplicação um único conjunto de dados. É válido salientar que o processador de consultas deverá realizar suas tarefas otimizando os custos da transferência de dados e do processamento. A arquitetura mostrando o processador de consultas é mostrado na figura 2.8 (ARAUJO DUARTE, 2004).

O processo de execução de uma consulta global é descrito por (ARAUJO DUARTE, 2004) da seguinte forma:

1. Uma consulta global é decomposta em subconsultas que são enviadas aos SGBDs locais. É importante salientar que o processamento das consultas nos SGBDs locais ocorre de forma paralela, ou seja, ao mesmo tempo.

2. Os SGBDs locais coordenam a execução das subconsultas e devolvem os dados obtidos para o processador de consultas.

3. O processador de consultas realiza a junção dos dados obtidos dos SGBDs componentes.

Essa abordagem de integração funciona, na concepção de (ARAUJO DUARTE, 2004), baseada na existência de uma camada de software localizada sobre os SGBDs

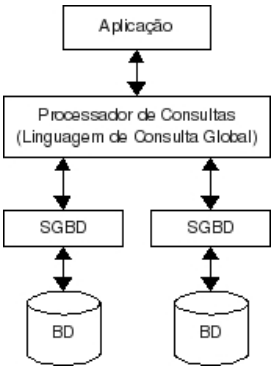

Figura 2.8: Arquitetura básica da integração baseada em linguagem de consulta.

locais, o processador de consultas, componente de um mediador, proporcionando aos usuários o acessos aos dados do BDH de forma transparente. Essa camada de software pode ser visualizada como o próprio SGBDH.

Essa abordagem será melhor detalhada no Capítulo 4, por ser base para a arquitetura do mediador que será proposta no Capítulo 5.

2.4 Considerações Finais

É possível identificar diversas vantagens na integração de bancos de dados distribuídos heterogêneos. Dentre as principais, pode-se destacar o compartilhamento e a disponibilidade das informações e o aumento da velocidade do processamento das consultas. Entretanto, conforme esperado de um sistema complexo como é o de integração, existem algumas desvantagens na sua construção e utilização. Dentre as principais desvantagens, merecem destaque o aumento do custo e a complexidade do desenvolvimento do software e o aumento do *overhead* no processamento.

2.4.1 Vantagens

São vantagens da integração de BDHs que merecem destaque:

1. **Compartilhamento das informações**: Com a existência de vários SGBDs locais que compõem o SGBDH, as informações são mais facilmente compartilhadas devido ao alto grau de distribuição dessas.

2. **Disponibilidade das informações**: Uma vez que todos os SGBDs locais estejam conectados, independentemente de sua localização no Planeta, é possível acessar suas informações estando em qualquer local, desde que o usuário tenha permissão para realizar o acesso e a conexão esteja ativa. Caso algum dos SGBDs locais esteja inoperante, ainda é possível acessar seus dados, desde que haja algum mecanismo de replicação das informações ativado.

2.4.2 Desvantagens

Dentre as desvantagens identificadas na integração de BDHs, pode-se citar:

1. **Aumento do custo e de complexidade do desenvolvimento do software**: Esse tipo de aplicação é de complexo desenvolvimento e, conseqüentemente, mais caro;

2. **Aumento do overhead no processamento**: As operações de integração podem exigir um grau mais alto de computação o que, juntamente com a comunicação em rede, pode ocasionar uma demora na obtenção dos resultados desejados.

É importante destacar o carater paradoxal da vantagem do aumento da velocidade no processamento das consultas e da desvantagem do aumento do *overhead* no processamento. Essa vantagem leva em consideração somente a execução de uma consulta global, desconsiderando todo o processo que existe antes e após a execução da consulta. Já a desvantagem considera todo o processo, desde a preparação do ambiente até o retorno dos dados ao usuário. Portanto, há ganho de tempo na execução da consulta e pode haver perdas no tempo de execução da integração como um todo.

3 MEDIADORES

De acordo com (WIEDERHOLD, 2007), mediadores são módulos de sistemas de informação que conectam várias fontes de dados às aplicações que necessitam acessá-las. Esses sistemas mediados são escalonáveis e possuem um algo grau de adaptação. São ditos escalonáveis por poderem ser ampliados conforme a necessidade e adaptáveis por ser possível mantê-los funcionando em ambientes de diversidade semântica e de rápidas mudanças. Os mediadores devem possuir conhecimento codificado sobre um conjunto ou subconjunto de dados (WIEDERHOLD, 1992). Neste caso, esse conjunto de dados é o conjunto sobre o qual o mediador irá trabalhar.

Os mediadores agem, segundo (BALLARDIN, 2004), como interfaces inteligentes que tratam os problemas de representação e abstração presentes nos BDHs. Ao contrário dos armazéns de dados, os mediadores trabalham com uma abordagem não materializada, ou seja, os dados não são extraídos dos BDs locais e armazenados em outro local. A extração e integração dos dados é feita sob demanda, obtendo-se sempre os dados atualizados dos BDs componentes.

Um mediador manipula um conjunto ou subconjunto de dados (ARAUJO DUARTE, 2004), podendo oferecer qualquer serviço relacionado à manipulação desses (ZANARDO, 1998). Esses serviços basicamente são oferecidos com a intenção de diminuir o volume de dados apresentado aos usuários, porém sem diminuir a quantidade de informação representada por esses dados. Neste ponto é importante ressaltar a diferenciação que ocorre entre dado e informação. Na visão de (WIEDERHOLD, 1992), dados são fatos armazenados relativos a alguma situação e informações são os resultados da interpretação dos dados pelo usuário, ou seja, o conhecimento que os dados fornecem ao indivíduo.

Dentre os diversos serviços que um mediador pode oferecer, na opinião de (ZANARDO, 1998), podem ser destacados:

- *Seleção de fontes de interesse*: Fontes de dados podem ser selecionadas por meio de palavras-chave informadas pelo usuário final do mediador ou por outro mediador do conjunto;

- *Execução e otimização de consultas globais*: O mediador pode realizar a otimização das consultas antes de executá-las, melhorando o desempenho do sistema. Nos bancos de dados relacionais, essa otimização geralmente é realizada utilizando-se a álgebra relacional[1];

- *Omissão de dados replicados*: O mediador realiza uma análise dos dados obtidos, retirando do conjunto retornado ao usuário, registros duplicados, caso existam;

- *Sumarização*: O mediador pode realizar a sumarização das informações vinculadas a um conjunto de dados.

- *Avaliação da qualidade das informações*: Com base em um conjunto pré-estabelecido de regras de avaliação, o mediador pode varrer diversos BDs avaliando a qualidade dos dados de cada um deles;

- *Imposição de filtros de segurança*: O mediador filtra as consultas dos usuários antes de executá-las, evitando, dessa forma, o acesso a informações que o usuário não possua permissão de acesso.

Os serviços apresentados podem ser implementados por um único mediador ou por vários mediadores interligados (ARAUJO DUARTE, 2004). Determinar quantos e quais serviços cada mediador irá implementar é uma tarefa complicada. Essa decisão deve considerar a complexidade e o relacionamento entre os serviços. Segundo (ZANARDO, 1998), a questão da decisão sobre os serviços de um mediador é chamada de *questão do escopo do mediador*.

Na opinião de (WIEDERHOLD, 1992), um mediador deve oferecer os serviços que digam respeito ao domínio dos dados. Considerando-se esse conceito, pode-se dizer então que um mediador possui duas dimensões, os serviços e o domínio. O primeiro diz respeito aos serviços que o mediador oferece, enquanto o segundo diz respeito ao domínio dos dados que serão manipulados, ou seja, o que esses dados representam no mundo real. Verificando-se a questão do escopo do mediador e suas dimensões, conclui-se que caso o mediador seja mal dimensionado sua construção pode tornar-se inviável.

De acordo com (ZANARDO, 1998), os problemas que podem ocorrer na definição do escopo de um mediador dizem respeito à escassez ou ao excesso de serviços e a complexidade desses. Tomando-se por premissas a quantidade e complexidade dos serviços e a extensão do domínio dos dados, pode-se afirmar, segundo

[1] A álgebra relacional é definida por (SILBERSCHATZ; KORTH; SUDARSHAN, 1999) como "uma linguagem de consultas que consiste em um conjunto de operações, tendo como entrada uma ou mais relações (tabelas) e produzindo, como resultado, uma nova relação".

(ARAUJO DUARTE, 2004), que essas devem ser equilibradas para não tornar o projeto de desenvolvimento demasiadamente caro e inviável.

3.1 Projetos desenvolvidos utilizando mediadores

A integração de BDHs é amplamente estudada (HARA et al., 2006). Essa sessão destina-se a explanar dois estudos que foram realizados e quais os resultados obtidos: HERMES e *J-Integrator*. Foram selecionados esses dois projetos porque ambos obtiveram sucesso no que se propuseram a realizar, sendo validados e utilizados nas instituições onde foram concebidos.

3.1.1 HERMES (*Heterogeneous Reasoning and Mediator System*)

O HERMES ou, traduzido do inglês, Sistema de Mediador e Raciocínio Heterogêneo, foi desenvolvido pelo Departamento de Ciência da Computação da Universidade de Maryland, Estados Unidos, com o apoio do Exército Americano (SUBRAHMANIAN et al., 1994). O HERMES objetiva integrar múltiplas fontes de dados e sistemas de raciocínio em um mesmo ambiente. Foram produzidas duas versões do sistema HERMES: uma para a plataforma IBM-PC, que funciona também sobre MS-DOS e Windows 3.1 e a outra para plataforma SUN/Unix que roda também sobre as versões mais recentes do Windows.

A arquitetura do HERMES é dividida basicamente em duas partes: construção do código do mediador e integração de novos domínios (SUBRAHMANIAN et al., 1994). Para realizar a integração dos novos domínios são utilizados mediadores cujas especificações são geradas no próprio ambiente HERMES através de uma linguagem de programação de mediadores e de um compilador. O sistema HERMES possui dois níveis de acesso: o autor de mediador e o usuário final. O autor de mediador é responsável pela criação dos mediadores e integração das fontes de dados enquanto os usuários finais acessam as interfaces de usuário que utilizam os mediadores criados.

O sistema HERMES foi aplicado pelo Exércido dos Estados Unidos no desenvolvimento de um sistema de raciocínio de terreno que integra um software de planejamento de trajetórias (SUBRAHMANIAN et al., 1994). Esse software utiliza vários BDs relacionais e espaciais para descrever aspectos da região coberta pelos mapas de terreno.

3.1.2 J-Integrator

O *J-Integrator* foi desenvolvido pelo Departamento de Informática da Universidade Federal do Paraná com o intuito de ser uma ferramenta de apoio a integração de esquemas de bancos de dados heterogêneos (HARA et al., 2006). Essa ferramenta foi dividida em duas partes, a primeira chamada *J-Schemata*, que realiza a

integração de esquemas e a outra, chamada *J-Query*, responsável pela execução de consultas globais na base de dados heterogênea.

O módulo *J-Schemata* é responsável, segundo (HARA et al., 2006), por auxiliar o usuário nos processos de identificação de correspondências, de mapeamento e de geração do ECG. Ao realizar essa integração de esquemas, a ferramenta gera um arquivo XML com a representação do ECG gerado.

O módulo *J-Query* funciona como um mediador. Ele é responsável por decompor a consulta do usuário com base no mapeamento XML gerado pelo *J-Schemata*, gerando as subconsultas que serão enviadas às fontes de dados (HARA et al., 2006). Após a execução da consulta, cabe ainda ao módulo *J-Query* realizar a junção dos dados obtidos para retornar o conjunto de dados ao usuário.

A ferramenta *J-Integrator* foi utilizada e validada na integração de algumas bases de dados da própia Universidade Federal do Paraná (HARA et al., 2006). Possíveis melhorias como, por exemplo, nas interfaces e otimização no processo de integração, foram identificadas como trabalhos futuros pelos pesquisadores e utilizadores da ferramenta.

3.2 Arquitetura de mediação em três níveis

Uma arquitetura de mediação é proposta por (WIEDERHOLD, 1992) e (WIEDERHOLD, 2007). Nessa arquitetura o sistema é implementado em três níveis. O nível intermediário, ou de mediação, localizado entre as aplicações dos usuários e os SGBDs, é responsável pela aquisição, manipulação e análise dos dados obtidos. No nível mais alto da arquitetura, chamado de nível de aplicação, ficam as aplicações que utilizam os dados obtidos e, no nível inferior, ou de dados, estão os BDs.

É importante observar que na arquitetura proposta por (WIEDERHOLD, 2007) e representada na Figura 3.1 as camadas podem ser formadas por mais de um de seus respectivos tipos de componentes. Na camada de mediação, pode-se ter um ou mais mediadores trabalhando conjuntamente. Na camada superior estão todas as aplicações dos usuários, relacionadas entre si ou não e, finalmente, na camada inferior estão os BDs que fornecem os dados.

De acordo com (WIEDERHOLD, 2007), os mediadores são responsáveis por serviços de extração e refinamento (conversões de tipos e filtragens, por exemplo) dos dados e, em alguns casos, processamento analítico desses. Para realizar essas tarefas os mediadores devem apresentar as características descritas a seguir, na visão de (WIEDERHOLD, 2007).

- *Possuir conhecimento*: O código de um mediador deverá encapsular algum tipo de conhecimento sobre o domínio dos dados;

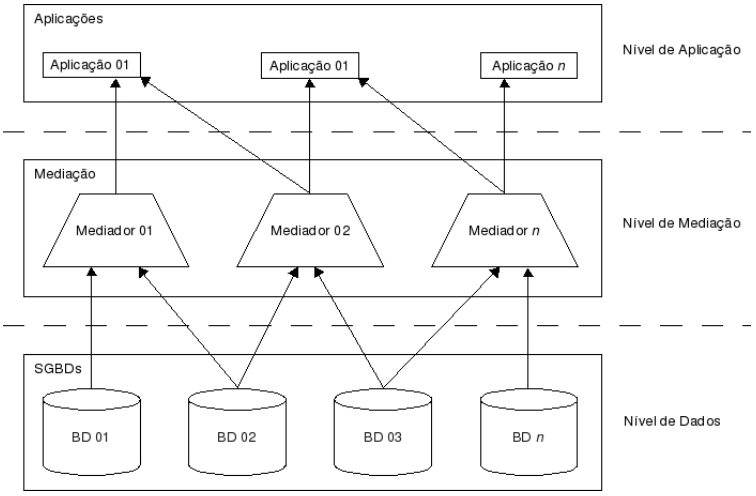

Figura 3.1: Exemplo de arquitetura de mediação.

- *Armazenar dados intermediários*: Um mediador deverá possuir um espaço para armazenamento temporário de informações, ou seja, um *buffer*. Esse *buffer* é utilizado para armazenar temporariamente os dados enquanto esses são refinados ou processados;

- *Fornecer serviços para o nível superior*: Um mediador é responsável por desempenhar algum papel específico devolvendo alguma informação para a camada superior, ou seja, a aplicação do usuário, ou então para um outro mediador;

- *Possuir informações sobre ele mesmo*: Um mediador deverá possuir informações sobre si próprio, ou seja, sobre os serviços por ele oferecidos.

- *Ser associativo*: Um mediador deverá ser capaz de interagir com outros mediadores formando, dessa maneira, uma cadeia de mediação e processamento.

Segundo (WIEDERHOLD, 2007), as aplicações que obtém dados do nível de mediação necessitam de informações refinadas e de qualidade. Cabe, portanto, à camada de mediação reduzir e refinar o volume de dados que será enviada à camada de aplicação. A subseção a seguir esclarece as funções que os mediadores implementam, de acordo com (WIEDERHOLD, 2007), para realizar suas tarefas reduzindo e refinando os dados.

3.2.1 Operações e Gerenciamento do Volume de Dados em Mediadores

Na arquitetura de mediador proposta por (WIEDERHOLD, 2007), a redução e refinamento dos dados é provida pelas seguintes operações:

1. Seleção dos dados relevantes a serem obtidos das fontes de dados.

2. Projeção, isto é, obtenção das colunas relevantes do conjunto de dados.

3. Agregação dos dados selecionados para obtenção do nível de abstração necessário.

4. Redução da integração e agregação dos dados menos relevantes.

5. Simplificação dos resultados para diminuir a carga das aplicações.

3.2.1.1 Seleção e Projeção

Conforme destacado por (WIEDERHOLD, 2007), a seleção (*SELECT*[2], em SQL) é a principal operação realizada em SGBDs. Obter somente os dados relevantes de um BD melhora significativamente a performance de uma aplicação, principalmente em casos em que há um grande volume de dados armazenados nesse BD.

A projeção diz respeito às colunas que serão recuperadas em uma consulta (WIEDERHOLD, 2007). É possível projetar todas ou somente algumas colunas de um conjunto de dados de um BD. Em BDs relacionais, a projeção pode reduzir também o número de linhas retornadas se alguns atributos chaves não forem projetados.

3.2.1.2 Agregação

De acordo com (WIEDERHOLD, 2007), fontes de dados diversas podem prover informações em diferentes níveis de abstração.

As funções de agregação (*COUNT*, *SUM*, *AVG*, *MAX*, *MIN*[3], em SQL) facilitam a abstração das informações e podem reduzir o volume de dados de entrada do mediador (WIEDERHOLD, 2007).

3.2.1.3 Abstração

O processo de abstração procura reduzir os dados a unidades mais significativas (WIEDERHOLD, 2007). A abstração geralmente é necessária antes da integração quando há diferenças na representação dos dados coletados de diversas fontes. Após o processo de integração, os dados podem ser agregados e abstraidos de acordo com as informações que os usuários necessitam.

3.2.1.4 Remoção de redundâncias após a integração

O processo de integração pode gerar alguma informação redundante (WIEDERHOLD, 2007). Obviamente, informações duplicadas devem ser omitidas, ou seja, não deve-

[2]Segundo (ELMASRI; NAVATHE, 2005), o comando *SELECT* utilizado para recuperar informações de um BD relacional.

[3]De acordo com (ELMASRI; NAVATHE, 2005), a função *COUNT* retorna o número de tuplas ou valores especificados em uma consulta. Já as funções *SUM*, *MAX*, *MIN* e *AVG* são responsáveis por retornar, respectivamente, a soma, o valor máximo, o valor mínimo e a média de um conjunto de valores.

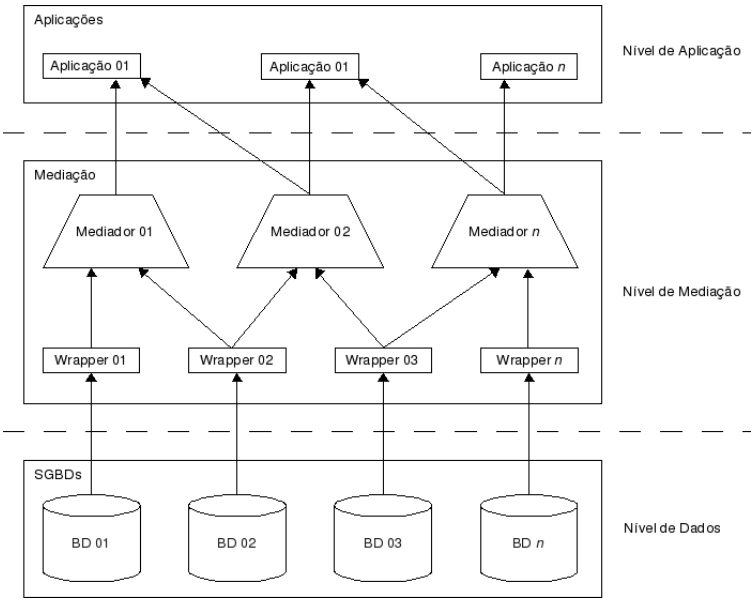

Figura 3.2: Arquitetura de mediação com a representação dos *wrappers*.

rão ser apresentadas para a camada superior, a camada de aplicação.

3.2.1.5 Classificação

A informação que será enviada para a camada de aplicação pode ser classificada e os dados que possuirem baixa classificação podem ser omitidos do resultado final ou então obtidos em uma consulta específica (WIEDERHOLD, 2007). A classificação dos dados pode reduzir de forma eficiente o volume de dados.

3.2.2 Conversores, Adaptadores ou *Wrappers*

Para seu funcionamento completo, a camada de mediação necessita de mais alguns componentes chamados conversores. Esses conversores, ou *wrappers*, são responsáveis pela conversão da linguagem de consulta do mediador na linguagem de consulta de cada fonte de informação (GARCIA-MOLINA; ULMAN; WIDOM, 2001). Além disso, cabe ainda aos *wrappers* a conversão dos dados obtidos da fonte de forma que o mediador os compreenda e consiga manipular.

A figura 3.2 exibe a localização dos *wrappers* na camada de mediação (WIEDERHOLD, 2007). Para cada SGBD ou outro tipo de fonte de dados que deva ser integrada, deverá haver um conversor específico (GARCIA-MOLINA; ULMAN; WIDOM, 2001).

Suponha uma fonte de dados manipulável através de uma linguagem 'X' e baseada em um modelo de representação de dados 'Y'. Imagine também um mediador que necessita acessar dados dessa fonte e utiliza para isso uma linguagem 'W' e um modelo de representação 'V'. Dessa forma, segundo (ARAUJO DUARTE, 2004), a

função de um *wrapper* é realizar a tradução de uma consulta na linguagem 'W' para a linguagem 'X'. Após a execução da consulta, o *wrapper* ainda deverá traduzir os dados obtidos do modelo de representação 'Y' para o modelo 'V', manipulado pelo mediador.

De acordo com (GRANVILLE, 1998), os *wrappers* ainda podem ser utilizados para prover mais funções além do mapeamento de linguagens e modelos envolvidos no sistema de mediação. Dentre essas funções, pode-se destacar as relacionadas a seguir.

- *Disponibilização de interfaces mais simples para aplicações clientes*: O *wrapper* encapsula a complexidade das consultas às fontes de dados. Caso a aplicação cliente não exija funcionalidades avançadas, o *wrapper* pode simplificar consultas escondendo eventuais complexidades da linguagem nativa da fonte de dados.

- *Disponibilização de interface padrão*: A camada de aplicação pode acessar diversas fontes de dados de forma padrão através de *wrappers* com interfaces idênticas.

- *Adição de funcionalidades às fontes de dados*: Os *wrappers* podem encapsular funções que originalmente não existem na fonte de dados a que estão ligados.

- *Exportação de interfaces das fontes de dados*: Algumas funções das fontes de dados acessíveis somente através dos *wrappers* podem ser indiretamente disponibilizadas para a camada de aplicação.

Não existe uma arquitetura padrão para o desenvolvimento dos *wrappers*. Para ajudar a definir a melhor arquitetura, algumas perguntas são colocadas para o desenvolvedor, dentre as quais, destacam-se (ARAUJO DUARTE, 2004):

1. Quais são as fontes de dados que o *wrapper* irá suportar?
2. Como será a interface externa?
3. Qual a função do *wrapper*?
4. Quais linguagens e modelos de representação serão suportados?
5. Por que as fontes de dados serão suportadas?
6. Como o *wrapper* será construído?
7. Como será a arquitetura interna do *wrapper*?

As perguntas de 1 a 4 servem como guia na seleção de um *wrapper* já existente ou na definição dos requisitos para o desenvolvimento de um novo (ARAUJO DUARTE, 2004). A pergunta 5 auxilia no dimensionamento da complexidade do *wrapper* e a sexta, na definição do paradigma de programação. Já a última pergunta auxilia na identificação dos módulos e funções, possibilitando o reuso posterior ou de módulos existentes.

3.2.3 Funcionamento da Camada de Mediação

Ao receber uma consulta da camada de aplicação, o mediador a decompõe conforme as fontes de dados que deverão ser consultadas (GARCIA-MOLINA; ULMAN; WIDOM, 2001). Essas subconsultas são, então, enviadas aos *wrappers* para que seja feita a conversão da linguagem de consulta do mediador para a linguagem de consulta da fonte à qual o *wrapper* está ligado (ARAUJO DUARTE, 2004). Feita essa conversão, as subconsultas são enviadas para as fontes e então executadas. Ao término da execução da subconsulta, o SGBD devolve o resultado ao *wrapper* que realiza a conversão dos dados do modelo da fonte para o modelo de dados do mediador (COSTA, 2005). O mediador, após receber os resultados das subconsultas, realiza a junção dos dados e os devolve à aplicação solicitante (GARCIA-MOLINA; ULMAN; WIDOM, 2001).

A figura 3.3 ilustra o funcionamento desse mecanismo de decomposição das consultas e tradução das mesmas na concepção de (COSTA, 2005). Na situação demonstrada, a consulta **Q** é enviada pela Aplicação para o Mediador. Esse por sua vez, realiza a decomposição desta nas subconsultas **Q1**, **Q2** e **Q3** que são enviadas para os seus respectivos *wrappers*. Após a conversão de linguagem das subconsultas, essas são enviadas aos SGBDs correspondentes e executadas. Os resultados gerados são enviados de volta aos *wrappers* para a tradução dos modelos de dados, gerando, dessa forma, os conjuntos de dados **Dados1**, **Dados2** e **Dados3**. Esses conjuntos de dados são enviados para o Mediador que realiza a sua junção em um único conjunto de dados que, finalmente, é enviado à Aplicação.

A estrutura do mecanismo de integração de BDHs através de mediadores é bastante simples. A complexidade desse mecanismo tende a aumentar com o detalhamento do funcionamento interno dos mediadores e dos *wrappers*, tanto nas operações de processamento para a execução das consultas, quanto na junção dos dados obtidos nos BDs componentes. Os mecanismos de decomposição e tradução das consultas contidos nos mediadores e *wrappers* serão apresentados no próximo capítulo, destinado ao processamento de consultas globais como forma de provimento de integração de BDHs. No capítulo 5 será realizado um fechamento de todos esses conceitos, propondo uma arquitetura para o desenvolvimento de um mediador e dos

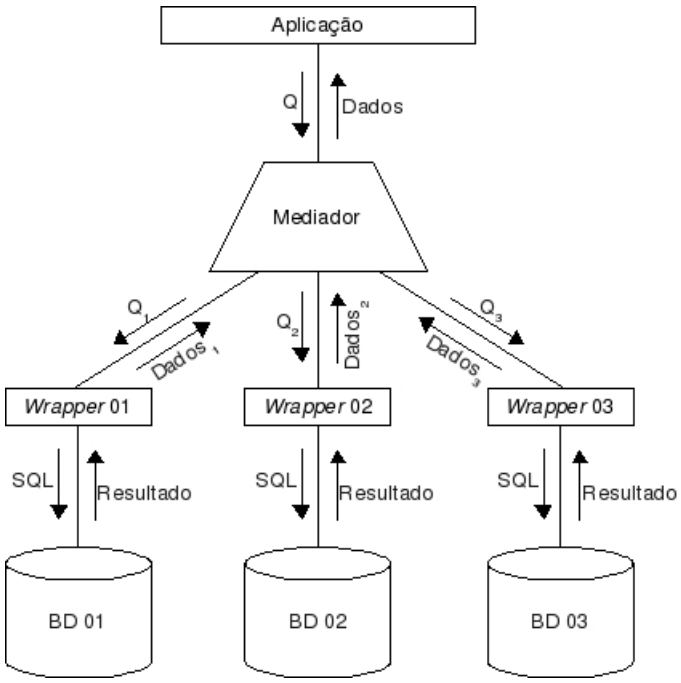

Figura 3.3: Funcionamento da Camada de Mediação.

wrappers para os SGBDs *PostgreSQL*[4] e *MySQL*[5].

[4]De acordo com (PostgreSQL, 2009), o PostgreSQL é "um poderoso SGBD objeto-relacional de código aberto". Possui mais de 15 anos de desenvolvimento e uma arquitetura confiável e íntegra, rodando em todos os grandes sistemas operacionais. Devido a todas essas vantagens, é um dos SGBDs mais utilizados no mundo (PostgreSQL, 2009).

[5]Segundo (MySQL, 2009), o MySQL é o SGBD de código aberto mais utilizado no mundo, devido ao fato de ser rápido, confiável e fácil de usar.

4 PROCESSAMENTO DE CONSULTAS GLOBAIS

Pode-se definir, de forma bastante genérica, o processamendo de consultas como o conjunto de atividades realizadas para a extração dos dados de um BD (ARAUJO DUARTE, 2004). Essas atividades abrangem desde a tradução das consultas expressas em linguagem de alto nível nas expressões que podem ser implementadas no nível físico dos computadores, otimizações e avaliações das mesmas. É válido ressaltar que essas atividades ocorrem independentemente de haver uma ou n consultas, mesmo que essas se originem de uma única consulta.

No contexto deste trabalho, o processamento de consultas será considerado como atividades realizadas pela camada de acesso aos dados. Essa camada é formada pelo mediador, pelos *wrappers* e pelos SGBDs que formam do SGBDH.

O processamento de consultas em SGBDHs é um problema abordado de diversas formas (LIMA; MELO, 1999). Uma solução, por exemplo, é a utilização de um SGBD que suporte um modelo de dados comum e uma linguagem de consulta global acima dos SGBDs componentes. Esse SGBD utiliza um esquema global resultante da integração dos esquemas locais.

Outra possível abordagem para consulta a dados de BDHs é a utilização de visões temporárias referentes às consultas dos usuários em substituição aos esquemas globais (LITWIN; MARK; ROUSSOPOULOS, 1990). Entretanto, nessa abordagem cabe à linguagem de consulta fornecer a possibilidade da criação das visões temporárias, ficando a cargo do usuário a criação e manutenção das visões criadas.

Segundo (LIMA; MELO, 1999), uma linguagem de consulta global deve ser utilizada para realizar consultas globais acessando esquemas globais. Uma consulta global é executada em três passos na proposta de (MENG; CLEMENT, 1995). Primeiramente a consulta global é decomposta em subconsultas de modo que cada sub-consulta obtenha os dados de um dos BDs componentes. Nessa fase as subconsultas ainda encontram-se na linguagem global. A tradução da linguagem global e seu envio para os SGBDs componentes ocorre somente no segundo passo. Após a execução da consulta, no terceiro passo, os dados obtidos são combinados, gerando um único conjunto de dados.

A linguagem de consulta global escolhida para a integração de BDHs deverá, segundo (ELMAGARMID; RUSINKIEWCZ; SHETH, 1999), possibilitar a realização das seguintes atividades:

1. *Manipulação de diversas tabelas*: Uma linguagem de consulta a vários BDs deverá possibilitar a manipulação simultânea de tabelas de esquemas distintos;

2. *Acesso a esquemas locais*: A linguagem de consulta global deverá ser capaz de acessar diretamente os esquemas dos BDs locais para realizar o mapeamento dos esquemas, quando for o caso de fazê-lo;

3. *Fornecimento transparente de localização*: Um nome de esquema global pode ser utilizado pela linguagem de consulta global para acesso aos dados;

4. *Acesso aos metadados:* O processador de consulta global deverá possuir acesso aos dicionários de dados dos BDs componentes;

5. *Resolução de conflitos*: A linguagem deverá prover algum mecanismo de resolução de conflitos semânticos em tempo de execução.

As etapas para a execução de uma consulta em BDHs apresentadas em (MENG; CLEMENT, 1995) são a base para a execução de uma consulta global. Entretando, existem distintas propostas para as atividades de cada etapa e a forma como essas são realizadas. A seguir será ilustrado o processo de execução da consulta conforme (MENG; CLEMENT, 1995) e também considerando as idéias apresentadas em (LITWIN; MARK; ROUSSOPOULOS, 1990). Foram selecionadas essas duas abordagens por abrangerem uma considerável parte das características do processamento de consultas nos mediadores.

4.1 Processamento de consultas em SGBDHs na visão de (MENG; CLEMENT, 1995)

Os SGBDHs realizam a tradução de consultas globais de forma adequada para cada SGBD local, realiza o envio das subconsultas para esses SGBDs e agrupam os resultados, gerando un único conjunto de dados para a aplicação ou diretamente para o usuário final (LIMA; MELO, 1999). Cabe ainda ao SGBDH a coordenação da execução da consulta global, ou seja, o cancelamento ou confirmação da execução da mesma.

O processamento de consultas globais em SGBDHs, segundo (MENG; CLEMENT, 1995), envolve as seguintes tarefas: decomposição, tradução e otimização global das consultas.

4.1.1 Decomposição da consulta

Quando há necessidade da execução de uma consulta global, essa é decomposta em dois tipos de consultas por um decompositor de consultas. O primeiro tipo é chamado sub-consulta do esquema enquanto o segundo, consulta pós-processamento.

Normalmente a decomposição de consultas é realizada em duas etapas: modificação e decomposição. Na etapa de modificação, os nomes de entidades e atributos globais são modificados para nomes que pertençam aos esquemas locais. Como esses nomes podem referenciar mais de um BD local, é necessário realizar a decomposição da consulta, ou seja, a divisão da consulta em subconsultas para o envio aos SGBDs componentes.

Existem alguns fatores dificultadores da etapa de modificação. Dentre esses fatores merecem destaque a linguagem de consulta global utilizada, o modelo de dados global, o método de integração de esquemas utilizado, inconsistência de dados e inconsistências semânticas.

4.1.2 Tradução da consulta

As subconsultas originadas da fase de decomposição são escritas na linguagem de consulta global, que pode não ser a mesma de algum dos SGBDs locais. Nessa fase essas subconsultas passam por um processo de tradução, transformando cada sub-consulta da linguagem global para a linguagem do SGBD local para a qual esta será enviada.

A complexidade desse processo de tradução é diretamente proporcional à complexidade da sintaxe e da expressividade das linguagens de origem e destino. Caso a linguagem de origem tenha maior poder de expressividade, ou seja, ofereça muitos recursos avançados, pode ocorrer que algumas consultas não possam ser traduzidas. Por exemplo, pode-se citar a realização de consultas recursivas em SGBDs orientados a objeto[1] que não podem ser traduzidas para consultas relacionais utilizando-se apenas SQL (LIMA; MELO, 1999).

4.1.3 Otimização global da consulta

O otimização das consultas globais visa a redução no tempo de processamento das mesmas. Esse processo de otimização em SGBDHs é diretamente ligada a otimização em SGBDs homogêneos. Entretanto, a aplicação dos mesmos algoritmos de otimização em ambos os casos é dificultada, por exemplo, pela inconsistência de dados e pela disponibilização de algumas informações dos SGBDs locais para o

[1] De acordo com (SILBERSCHATZ; KORTH; SUDARSHAN, 1999), o paradigma de orientação a objetos baseia-se no encapsulamento de dados e do código relacionado a um objeto dentro de una única unidade. Em comparação ao modelo relacional, um objeto representa uma entidade desse modelo. Logo, os SGBDs orientados a objeto diferem dos SGBDs relacionais por armazenarem esses objetos na sua estrutura própria ao invés da estrutura relacional.

otimizador global. Dentre essas informações, pode-se destacar as cardinalidades e a seletividade.

Na tentativa de solucionar a falta de informações dos SGBDs locais para a otimização, pode-se enviar, previamente, consultas de amostragem para os BDs locais. É importante ressaltar que, caso essa solução seja adotada, haverá aumento no custo da otimização, pois deve-se considerar o tempo de realização dessas consultas de amostragem, uma vez que as mesmas são sempre executadas antes das consultas globais.

4.2 Processamento de consultas em SGBDHs na visão de (LITWIN; MARK; ROUSSOPOULOS, 1990)

As etapas do processamento de consultas em SGBDHs propostas por (LITWIN; MARK; ROUSSOPOULOS, 1990) diferem das propostas por (MENG; CLEMENT, 1995) em alguns aspectos. Na proposta de (LITWIN; MARK; ROUSSOPOULOS, 1990), as etapas de processamento de uma consulta global são: formulação de consultas, tranformação de comandos e processamento e otimização das consultas.

4.2.1 Formulação de consultas

Nessa etapa, uma única linguagem pode ser utilizada para formulação da consulta global. Com essa linguagem, pode-se definir visões sobre os esquemas locais e, posteriormente, consultas sobre essas visões locais. Além disso, a linguagem global ainda considera os problemas de integração como conflitos de nome e estruturas de dados.

4.2.2 Transformação de comandos

Na segunda etapa, são realizadas atividades de transformação de comandos, ou seja, a tradução dos comandos de uma linguagem fonte, no caso a linguagem de consulta global, em uma linguagem destino, as linguagens dos SGBDs locais.

4.2.3 Processamento e otimização das consultas

O processamento das consultas diz respeito à conversão da consulta realizada ao esquema global em termos dos esquemas locais.

Quanto à otimização, são válidas as mesmas idéias de (MENG; CLEMENT, 1995), salientando-se ainda que o custo para realizar um consulta é afetado pela autonomia de cada SGBD componente.

	Meng	Litwin
Primeira etapa	A decomposição da consulta é realizada em duas fases e gera dois tipos de subconsultas. Uma dessas fases é a própria decomposição. Na outra fase, os nomes de entidades e atributos do ECG são substituídas pelos nomes de entidades e atributos dos esquemas locais.	A formulação das subconsultas é baseada em visões sobre os esquemas locais dos BDs componentes.
Segunda etapa	Nesta etapa, em ambas as propostas, as subconsultas são traduzidas da linguagem de consulta global para as linguagens implementadas pelos SGBDs locais.	
Terceira etapa	O objetivo da etapa de otimização é reduzir o tempo de execução da consulta global como um todo.	Além dos processos de otimização, nesta etapa, os nomes de entidades e atributos do ECG são substituídos pelos nomes de entidades e atributos dos esquemas locais.

Tabela 4.1: Resumo e comparação das abordagens de Meng e Litwin.

4.3 Comparação das abordagens de (MENG; CLEMENT, 1995) e (LITWIN; MARK; ROUSSOPOULOS, 1990)

A tabela 4.1 aponta algumas sutis diferenças existentes entre as abordagens estudadas considerando as principais operações do processamento de consultas: decomposição, tradução e otimização da consulta global e substituição dos nomes das entidades e atributos do ECG pelos nomes de entidades e atributos dos esquemas locais.

4.4 Processamento de consultas com o uso de mediadores

O processamento de consultas globais em SGBDHs com o uso de mediadores reúne características de ambas idéias de processamento apresentadas na sessão anterior. Segundo (LIMA; MELO, 1999), para realizar o acesso uniforme a múltiplas fontes de dados utilizando um mediador, este deve aceitar as consultas em uma linguagem de consulta global e decompô-la em subconsultas que são enviadas aos SGBDs componentes. Após cada sub-consulta ser processada nos SGBDs locais, os dados obtidos são retornados para o mediador, que os une e devolve à aplicação.

Os mediadores necessitam, para a realização das conversões de dados e traduções

das linguagens, do auxílio dos *wrappers*, cuja estrutura e funcionamento foram apresentados nas subsessões 3.2.2 e 3.2.3, respectivamente. De forma resumida, pode-se dizer que os *wrappers* são responsáveis, de acordo com (LIMA; MELO, 1999), por transformar as consultas em linguagem de consulta global para a linguagem dos SGBDs locais. Além disso, os *wrappers* são responsáveis por reformatar as respostas obtidas dos BDs locais para a representação de dados do mediador.

4.5 Arquitetura do processador de consulta

As consultas globais referenciam entidades e atributos localizados no esquema global (SMITH et al., 1986). Antes de serem executadas, é necessário realizar a tradução das consultas globais em termos das entidades e atributos dos esquemas locais. De forma mais simples, pode-se dizer que a tarefa do processador de consultas é traduzir uma consulta global sobre um esquema global em subconsultas sobre os esquemas locais a que se referem.

A figura 4.1 mostra, na visão de (SMITH et al., 1986), a arquitetura do processador de consultas que é utilizado na construção de mediadores. Essa arquitetura é composta basicamente pelo tradutor e pelo processador de consultas. Os demais componentes pertencentes ao processador de consultas do mediador são os *wrappers*, que já foram explanados no capítulo 3.

O processamento das consultas no mediador proposto por (SMITH et al., 1986) é, em suma, o mesmo proposto por (MENG; CLEMENT, 1995). Os componentes do processador de consultas e seu inter-relacionamento são explanados a seguir.

4.5.1 Tradutor de consultas

O tradutor de consultas recebe as consultas globais em termos do esquema global (SMITH et al., 1986), conforme é possível verificar na figura 4.1. Para realizar a tradução, o tradutor utiliza o mapeamento das entidades e atributos dos BDs locais, ou seja, o ECG.

O tradutor de consultas utiliza o ECG para substituir as entidades e atributos globais da consulta global pelas entidades e atributos dos esquemas locais.

4.5.2 Processador de consultas

O processador de consultas é responsável, segundo (SMITH et al., 1986), pela decomposição da consulta global nas subconsultas que serão enviadas para os SGBDs componentes. Essas subconsultas operam sobre os esquemas locais dos BDs componentes. O funcionamento do processador de consultas em mediadores, segundo (SMITH et al., 1986), é mostrado na figura 4.1.

O processamento de consultas nessa fase, ocorre pela traduçao interna da con-

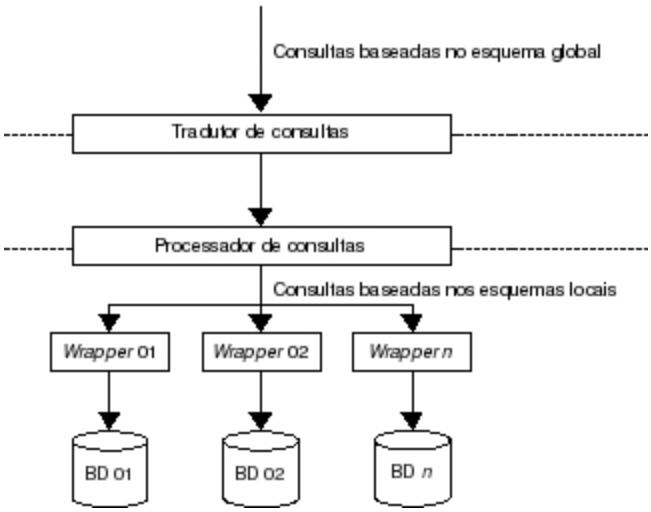

Figura 4.1: Funcionamento do processador de consultas em mediadores.

Figura 4.2: Esquema da base de dados da revenda A.

sulta em um grafo de consulta. Com base nesse grafo, o processador de consultas isola as subconsultas que serão enviadas aos SGBDs componentes.

4.6 Exemplo de processamento de consultas

Para exemplificar como uma consulta global é decomposta e traduzida, será considerada a situação hipotética de uma revenda de carros proposta por (GARCIA-MOLINA; ULMAN; WIDOM, 2001).

Nessa situação existe um sistema que necessita acessar duas bases de dados de revendas diferentes. A figura 4.2 mostra o esquema da base de dados da revenda A enquanto a figura 4.3 exibe o esquema do BD da revenda B (GARCIA-MOLINA; ULMAN; WIDOM, 2001).

Supondo que a aplicação que acessa esses dois BDs necessite saber quais são os carros vermelhos que possuem CD-Player que estão nas revendas. A seguinte consulta, em uma linguagem global poderia ser enviada ao mediador:

```
SELECT nr_serie, modelo FROM automoveis
WHERE nm_cor = 'Vermelho' AND ind_cd_player = 1;
```

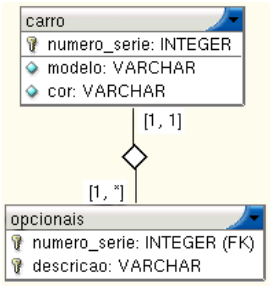

Figura 4.3: Esquema da base de dados da revenda B.

Revenda A	**Revenda B**

`SELECT nr_serie, nm_modelo`
`FROM carro`
`WHERE nm_cor = 'Vermelho'`
`AND ind_cd_player = 1;`
	`SELECT numero_serie, modelo`
	`FROM carro c, opcionais o`
	`WHERE c.numero_serie = o.numero_serie`
	`AND cor = 'Vermelho';`
	`AND descricao = 'CD-Player';`

Tabela 4.2: Consultas enviadas aos BDs das revendas A e B.

Ao receber essa consulta o mediador realizará a decomposição e a tradução das subconsultas geradas. O *wrapper* para o BD da revenda A realiza a conversão da sua sub-consulta para os termos da linguagem e do esquema do BD A. O mesmo ocorre no *wrapper* para o BD da revenda B. Um exemplo de como são essas subconsultas pode ser visualizado na tabela 4.2.

Após a execução das subconsultas nos SGBDs componentes, os resultados obtidos são retornados para o *wrapper*, que realiza a conversão dos tipos de dados do seu respectivo SGBD local para os tipos de dados do mediador. O mediador, ao receber esses conjuntos de dados, ou seja, os números de série e modelos dos carros vermelhos com CD-Player, realiza a junção dos mesmos, retornando à aplicação um único conjunto de dados (GARCIA-MOLINA; ULMAN; WIDOM, 2001).

5 PROPOSTA DE ARQUITETURA DE MEDIADOR BASEADO EM CONSULTAS GLOBAIS

Como foi possível observar até aqui, as propostas de funcionamento dos mediadores possuem basicamente a mesma estrutura. Em termos de arquitetura, pode-se dizer que todas as propostas de mediadores estudadas são equivalentes, sofrendo somente adaptações conforme o estudo em que estão inseridas.

Deste modo, este capítulo apresentará uma proposta de arquitetura para a implementação de um protótipo de um mediador. Essa arquitetura será proposta de forma que seja a mais abrangente possível, podendo atender a maioria dos SGBDs relacionais disponíveis atualmente, que utilizam a linguagem SQL padrão. Para a proposta dessa arquitetura serão consideradas as melhores, mais importantes e exeqüíveis idéias de cada proposta estudada, dimensionando o escopo da mesma para a implementação que será detalhada no capítulo 6.

5.1 Escopo do mediador

O mediador aqui proposto tem por objetivo funcionar como um componente de acesso a dados de diversos BDs. Tal componente destina-se ao desenvolvimento de aplicações na linguagem Java. Dentre as principais características que serão explanadas, pode-se enfatizar a possibilidade de uma única consulta ser executada em n fontes de dados (BDs locais). Essas n fontes de dados se apresentarão como uma única fonte ao desenvolvedor da aplicação, ficando o acesso individual a cada BD local encapsuldado no mediador e transparente a este.

O aplicativo de simulação, que pode ser observado na camada de aplicação da figura 5.1, funcionará como um executor de consultas nas bases de dados utilizadas para a simulação. Nessa aplicação, serão escritas as consultas globais que, quando submetidas ao mediador, darão origem às subconsultas que serão exibidas na interface da aplicação, juntamente com os dados obtidos nas bases de dados. Os dados obtidos nos BDs locais serão exibidos separadamente e unificados. Não é o foco deste trabalho o estudo do processo de junção dos dados obtidos, portanto, o pro-

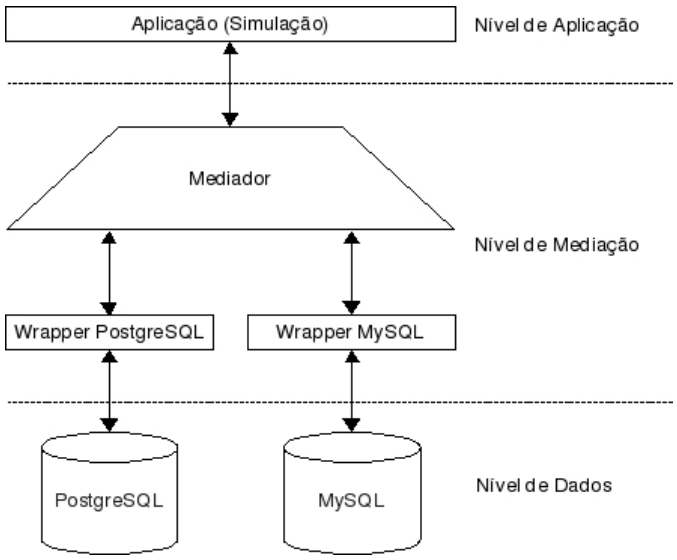

Figura 5.1: Escopo da arquitetura de mediador e aplicação de simulação.

tótipo de mediador proposto somente realizará a união dos dados obtidos, omitindo os registros duplicados.

As bases de dados utilizadas na simulação são bases públicas, disponibilizadas pelo Governo dos Estados Unidos. Estas bases contêm informações sobre estimativas populacionais dos estados norte-americanos nos anos de 2007 e 2008, baseadas nos censos realizados desde o ano 2000. Os dados estimados referentes a cada ano foram importados para os SGBDs utilizados nas simulações, PostgreSQL e MySQL. Portanto, os *wrappers* a serem desenvolvidos serão específicos para esses SGBDs.

Além dos *wrappers*, será desenvolvido o mediador cujo foco é o processamento de consultas globais. Essas consultas globais, contendo entidades e atributos do ECG, serão decompostas e traduzidas pelo mediador para que cheguem aos *wrappers* contendo somente entidades e atributos dos seus respectivos esquemas locais. A cada *wrapper* caberá somente a tradução da linguagem de consulta global para a linguagem de consulta utilizada pelo seu respectivo SGBD, quando houver necessidade.

A arquitetura de mediação proposta para o desenvolvimento está baseada na arquitetura proposta por (WIEDERHOLD, 2007) e (GARCIA-MOLINA; ULMAN; WIDOM, 2001), que pode ser visualizada na figura 3.2. A figura 5.1 exibe a mesma arquitetura adaptada para a realidade do desenvolvimento deste protótipo de mediador.

O mediador, os *wrappers* e a aplicação de simulação serão desenvolvidas em linguagem Java devido ao fato dessa ser uma das linguagens mais utilizadas no planeta, além de apresentar outras vantagens como, por exemplo, ser independente de plataforma e de código aberto.

Comando / Cláusula	Descrição
SELECT lista_de_atributos	Comando utilizado para recuperar informações armazenadas em um BD relacional. A lista_de_atributos contém os campos que deverão ser retornados pela consulta.
FROM lista_de_tabelas	Cláusula utilizada para informar quais tabelas serão utilizadas no processamento da consulta.
JOIN nome_tabela condição	Cláusula opcional utilizada para realizar a junção entre tabelas. Serão admititas as junções do tipo *INNER*, *LEFT*, *RIGHT* e *FULL* previstas no padrão SQL-99.
WHERE condições	Cláusula opcional que contém uma expressão condicional que identifica quais tuplas serão recuperadas pela consulta.

Tabela 5.1: Comandos aceitos pelo protótipo do mediador.

Operador	Descrição
AND	Operador lógico E.
OR	Operador lógico OU.
NOT	Operador lógico NÃO.
<	Operador relacional menor.
<=	Operador relacional menor ou igual.
>	Operador relacional maior.
>=	Operador relacional maior ou igual.
<>	Operador relacional diferente.
=	Operador relacional igual.

Tabela 5.2: Operadores lógicos e relacionais aceitos pelo protótipo do mediador.

5.2 Comandos de consulta a serem abordados

O padrão SQL-99 será utilizado como linguagem de consulta global por ser suportado pela grande maioria dos bancos de dados atualmente disponíveis, por possuir recursos adicionais como, por exemplo, gatilhos, e por suportar maior quantidade de tipos de dados (ELMASRI; NAVATHE, 2005).

Entretanto, o padrão SQL-99 prevê muitas opções para os comandos de consultas (ANSI-SQL, 1999). Para o escopo de desenvolvimento deste trabalho, será utilizada uma versão simplificada do comando de consulta previsto no padrão.

Os comandos e cláusulas listados na tabela 5.1 serão tratados pelo protótipo de mediador desenvolvido. Junto ao comando também é apresentada uma breve descrição de sua função conforme (ELMASRI; NAVATHE, 2005).

A cláusula *FROM*, segundo o padrão SQL-99 (ANSI-SQL, 1999), admite que seja

Operador	Descrição
LIKE	Operador utilizado para comparações de partes de uma cadeia de caracteres. Esse operador pode ser utilizado juntamente com dois caracteres curingas: '%' - utilizado para substituir uma quantidade qualquer de caracteres - e '_' - utilizado para substituir um único caracter.
BETWEEN	Operador de comparação utilizado para o teste de intervalos de valores.
IN	Operador de comparação que compara um valor qualquer com um conjunto dado de valores.
IS NULL	Operador de comparação utilizado para testar se um determinado valor é nulo ou não.
||	Operador de concatenação de cadeias de caracteres.
+	Operador de soma utilizado na definição de expressões aritméticas.
-	Operador de subtração utilizado na definição de expressões aritméticas.
*	Operador de multiplicação utilizado na definição de expressões aritméticas.
/	Operador de divisão utilizado na definição de expressões aritméticas.

Tabela 5.3: Demais operadores aceitos pelo protótipo do mediador.

informada mais de uma tabela para ser utilizada no processamento da consulta[1] ou então a utilização do comando *JOIN* para realizar junções entre tabelas. No escopo do desenvolvimento do mediador aqui proposto, não será admitida a junção ou a utilização de tabelas de diferentes BDs locais na mesma Consulta Global. Entretanto, caso as junções ou o produto cartesiano ocorram entre tabelas de um mesmo BD local, o mediador retornará sem problemas os resultados obtidos.

A limitação da localização da tabelas utilizadas nos produtos cartesianos ou junções deve-se ao fato de que, para retornar o resultado obtido correto para o usuário da aplicação, é necessário realizar o processamento e a junção dos dados após a execução das consultas locais, o que acaba fugindo ao escopo deste trabalho.

As condições utilizadas junto às cláusulas opcionais *WHERE* e *JOIN* devem ser expressões condicionais, ou seja, *booleanas*, retornando como resultado os valores VERDADEIRO ou FALSO. Na tabela 5.2 são mostrados os operadores lógicos e relacionais que serão suportados pelo protótipo de mediador.

Existem outros operadores que podem ser utilizados na cláusula *WHERE*: *LIKE*, *BETWEEN*, *IN*, *IS NULL*. Além destes, existem operadores que podem ser utilizados na junção de cadeias de caracteres (|| - concatenação) e na definição de expressões aritméticas (+, -, * e /). As funções destes operadores são explanadas na tabela 5.3, com base nas idéias de (ELMASRI; NAVATHE, 2005), uma vez que os mesmos serão suportados pelo protótipo de mediador.

É válido ressaltar que, no escopo deste trabalho, o operador *IN* aceitará somente conjuntos de valores previamente dados, não sendo aceitas consultas aninhadas. Consultas aninhadas, ou seja, consultas dentro de outras consultas, necessitam de parte do processamento da junção dos dados, fugindo ao foco deste trabalho.

As demais opções definidos no padrão SQL-99 não serão tratados por serem melhor aplicados no momento do processamento da junção das informações obtidas dos BDs componentes ou porque, para seu correto funcionamento, necessitem de uma junção parcial dessas informações. Logo, não farão parte deste protótipo, por exemplo, as seguintes funcionalidades: agrupamento e funções de agregação, ordenação e operações sobre conjuntos (intersecção, união e diferença). Além dessas, mesmo baseadas em consultas, não será tratado a definição de visões sobre BDs, uma vez que não é este o foco deste trabalho.

A seguir pode-se visualizar a BNF simplificada da consulta global suportada pelo mediador de acordo com o padrão SQL-99 (ANSI-SQL, 1999).

```
<consulta_global> ::= SELECT <lista_select> <clausula_from>
<lista_select> ::= id_attributo [{, id_atributo}...]
```

[1]Segundo (SILBERSCHATZ; KORTH; SUDARSHAN, 1999), define-se a operação de produto cartesiano como uma forma de combinação entre as tuplas de duas tabelas. Nesta combinação, cada linha da primeira tabela é relacionada com todas as linhas da segunda tabela da operação.

```
<clausula_from> ::= FROM id_tabela [<lista_join>] [<clausula_where>]
<lista_join> ::= <clausula_join> [{<clausula_join>}...]]
<clausula_join> ::= [<tipo_join>] JOIN id_tabela ON condicao
<tipo_join> ::= INNER | <tipo_outer_join> [OUTER]
<tipo_outer_join> ::= LEFT | RIGHT | FULL
<clausula_where> ::= WHERE condicao
```

5.3 Processamento de consultas

O processamento de consultas é responsável por grande parte do custo da integração de BDHs (ARAUJO DUARTE, 2004), ficando o processador de consultas, localizado no mediador.

Para o desenvolvimento do mediador, as etapas deste processamento propostas por (MENG; CLEMENT, 1995) e (SMITH et al., 1986) serão divididas em seis passos distintos, seguindo o estudo de (ARAUJO DUARTE, 2004): identificação dos elementos da consulta, localização dos dados, decomposição da consulta e envio das subconsultas.

5.3.1 Identificação dos elementos da consulta

Nesta etapa o mediador identificará os elementos da consulta global a ser processada, identificando quais os dados que irão compor o resultado da consulta. Como linguagem global será utilizado o padrão SQL-99, implementado pela grande maioria dos SGBDs atuais.

5.3.2 Localização dos dados

O mediador realiza uma busca no ECG, do mapeamento de cada elemento da consulta global, retornando seus respectivos elementos locais dos BDs locais. Nesta etapa o mediador relacionará cada elemento global com o seu respectivo elemento local, podendo esse estar localizado em mais de um BD componente.

5.3.3 Decomposição da consulta

Nesta etapa, com base nos resultados das etapas anteriores, é realizada a decomposição da consulta, gerando as subconsultas que serão enviadas aos *wrappers*.

5.3.4 Envio das subconsultas

Após a geração das subconsultas, as mesmas são enviadas aos *wrappers* de cada SGBD local, onde são processadas.

5.3.5 União dos resultados obtidos

Após o processamento das subconsultas nos SGBDs locais, os *wrappers* irão devolver ao mediador os conjuntos de valores obtidos. Esses valores precisam ser unificados pelo mediador em um único conjunto que será retornado para a aplicação do usuário.

5.3.6 Visualização dos resultados obtidos

Após a união dos resultados obtidos, o conjunto unificado de dados é retornado para a aplicação do usuário para que essa realize as operações que se fizerem necessárias sobre estes.

Existe ainda mais uma etapa do processamento de consultas que é executada por cada *wrapper* que recebe uma subconsulta. Nesta etapa, a linguagem da subconsulta, que até o momento é a linguagem global, é convertida para a linguagem do SGBD ao qual o *wrapper* está ligado (GARCIA-MOLINA; ULMAN; WIDOM, 2001).

O controle da execução das consultas globais dar-se-á por transações globais. Entretanto, por não ser este o foco deste trabalho, uma estratégia simplificada de controle dessa transação global será apresentada no capítulo 6, que realiza o detalhamento da implementação do protótipo de mediador proposto.

5.4 Serviços oferecidos pelo protótipo mediador

Um mediador pode oferecer quaisquer serviços que dizem respeito a manipulação dos dados. No capítulo 03 foram apresentados alguns desses possíveis serviços. Por não serem o foco deste estudo as técnicas de junção das informações obtidas, o protótipo de mediador aqui proposto oferecerá como serviço a união dos dados obtidos em um único conjunto de dados, omitindo-se os registros duplicados. Esse novo conjunto de dados é, então, retornado pelo mediador para a aplicação do usuário.

5.5 Princípios para utilização do mediador proposto

O protótipo de mediador aqui proposto utilizará um ECG das bases de teste previamente definido, pois não é o foco deste trabalho a integração ou o mapeamento dos esquemas. Tal ECG estará representado num arquivo XML.

Para exemplificar este mapeamento em XML, será utilizada a figura 5.2, que contém a representação dos esquemas locais de dois BDs hipotéticos.

Como forma de representação do ECG gerado a partir dos dois esquemas locais do exemplo, foi gerado o seguinte arquivo XML (ARAUJO DUARTE, 2004). É importante destacar que o mesmo modelo XML para o ECG do exemplo será utilizado

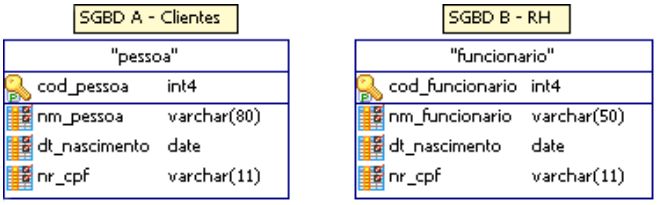

Figura 5.2: Esquemas locais dos BDs hipotéticos para exemplificação do mapeamento XML.

no desenvolvimento e utilização do protótipo do mediador.

```
01. <?xml version="1.0" encoding="UTF-8"?>
02. <?xmlspysps Schema.sps?>
03. <EsquemaGlobal>
04.     <Elementos>
05.         <Elemento>
06.             <CampoGlobal nome = "codigo_pessoa">
07.                 <Banco nome="postgresql">
08.                     <Tabela>pessoa</Tabela>
09.                     <Campo>cod_pessoa</Campo>
10.                 </Banco>
11.                 <Banco nome="mysql">
12.                     <Tabela>funcionario</Tabela>
13.                     <Campo>cod_funcionario</Campo>
14.                 </Banco>
15.             </CampoGlobal>
16.         </Elemento>
17.         <Elemento>
18.             <CampoGlobal nome = "nome_pessoa">
19.                 <Banco nome="postgresql">
20.                     <Tabela>pessoa</Tabela>
21.                     <Campo>nm_pessoa</Campo>
22.                 </Banco>
23.                 <Banco nome="mysql">
24.                     <Tabela>funcionario</Tabela>
25.                     <Campo>nm_funcionario</Campo>
26.                 </Banco>
27.             </CampoGlobal>
28.         </Elemento>
29.         <Elemento>
```

```
30.             <CampoGlobal nome = "nascimento_pessoa">
31.                 <Banco nome="postgresql">
32.                     <Tabela>pessoa</Tabela>
33.                     <Campo>dt_nascimento</Campo>
34.                 </Banco>
35.                 <Banco nome="mysql">
36.                     <Tabela>funcionario</Tabela>
37.                     <Campo>dt_nascimento</Campo>
38.                 </Banco>
39.             </CampoGlobal>
40.         </Elemento>
41.         <Elemento>
42.             <CampoGlobal nome = "cpf_pessoa">
43.                 <Banco nome="postgresql">
44.                     <Tabela>pessoa</Tabela>
45.                     <Campo>nr_cpf</Campo>
46.                 </Banco>
47.                 <Banco nome="mysql">
48.                     <Tabela>funcionario</Tabela>
49.                     <Campo>nr_cpf</Campo>
50.                 </Banco>
51.             </CampoGlobal>
52.         </Elemento>
53.     </Elementos>
54. </EsquemaGlobal>
```

Existem alguns elementos representados neste exemplo de XML que merecem destaque: o nome do BD, o nome da tabela e os nomes dos campos dos BDs locais e o nome dos elementos do ECG. O bloco representado entre as linhas 05 e 16 representa um elemento do ECG. Esse elemento representa um campo do ECG, no caso do exemplo, chamado *codigo_pessoa*. Esse campo do ECG é mapeado em dois campos dos BDs locais. No bloco formado pelas linhas de 07 a 10 é representado um desses BDs, cujo nome pode ser encontrado na linha 07. Nas linhas 08 e 09, respectivamente, são encontrados os nomes da tabela e do campo do BD local que é mapeado pelo elemento em questão.

5.6 Considerações Finais

A arquitetura de mediador proposta neste trabalho não realizará o processamento da junção dos dados obtidos das fontes. Tampouco os *wrappers* realizarão a conversão explícita dos modelos de dados dos SGBDs para um modelo de dados canônico, ou seja, único, o qual o mediador seja capaz de manipular para realizar essa junção. Essa heterogeneidade dos modelos de dados dos SGBDs é eliminada pela utilização dos *drivers* JDBC, desde que esses sejam implementados conforme o padrão (Sun Microsystems, 2006).

A proposta aqui apresentada preocupa-se somente com um único sentido da comunicação para extração de dados de um BDH. Somente a preparação para a execução das consultas é foco deste trabalho, sendo o sentido inverso dessa comunicação, ou seja, a obtenção e processamento dos resultados obtidos, objetos para outros estudos futuros. Entretanto, o protótipo proposto realizará um processo de união simplificado dos resultados para que seja possível visualizar os resultados obtidos na aplicação de simulação.

6 DETALHAMENTO DO DESENVOLVIMENTO DO PROTÓTIPO DE MEDIADOR

Com base na arquitetura básica definida no capítulo anterior, foi realizado o desenvolvimento do protótipo de mediador e da aplicação de simulação que utiliza o mesmo. Neste capítulo será detalhada a arquitetura utilizada e desenvolvida, bem como a integração entre os diversos componentes que formam o protótipo de mediador.

Os componentes internos mediador desenvolvido foram divididos e agrupados conforme suas funções. Os pacotes gerados estão presesentados na figura 6.1. Nas próximas seções esses pacotes serão detalhados, mostrando quais classes os compõem, suas principais estruturas internas e funções. Na figura 6.2 pode ser visualizado o digrama de classes do protótpio de mediador desenvolvido.

6.1 Pacote analisador

Neste pacote estão agrupadas as classes responsáveis por realizar a análise léxica da consulta global submetida ao mediador. As classes pertencentes a esse pacote são *AnalisadorLexico* e *Lexema*.

De acordo com (ALENCAR PRICE; TOSCANI, 2001), o processo de análise

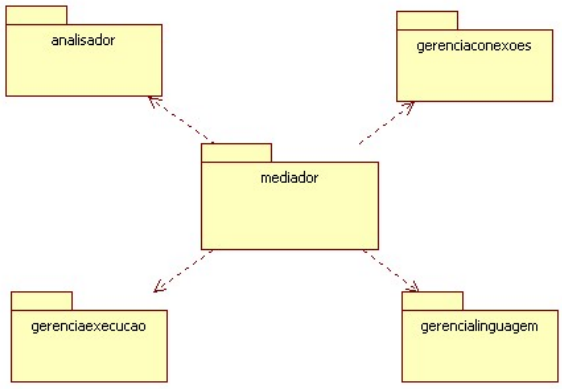

Figura 6.1: Pacotes do protótipo de mediador desenvolvido.

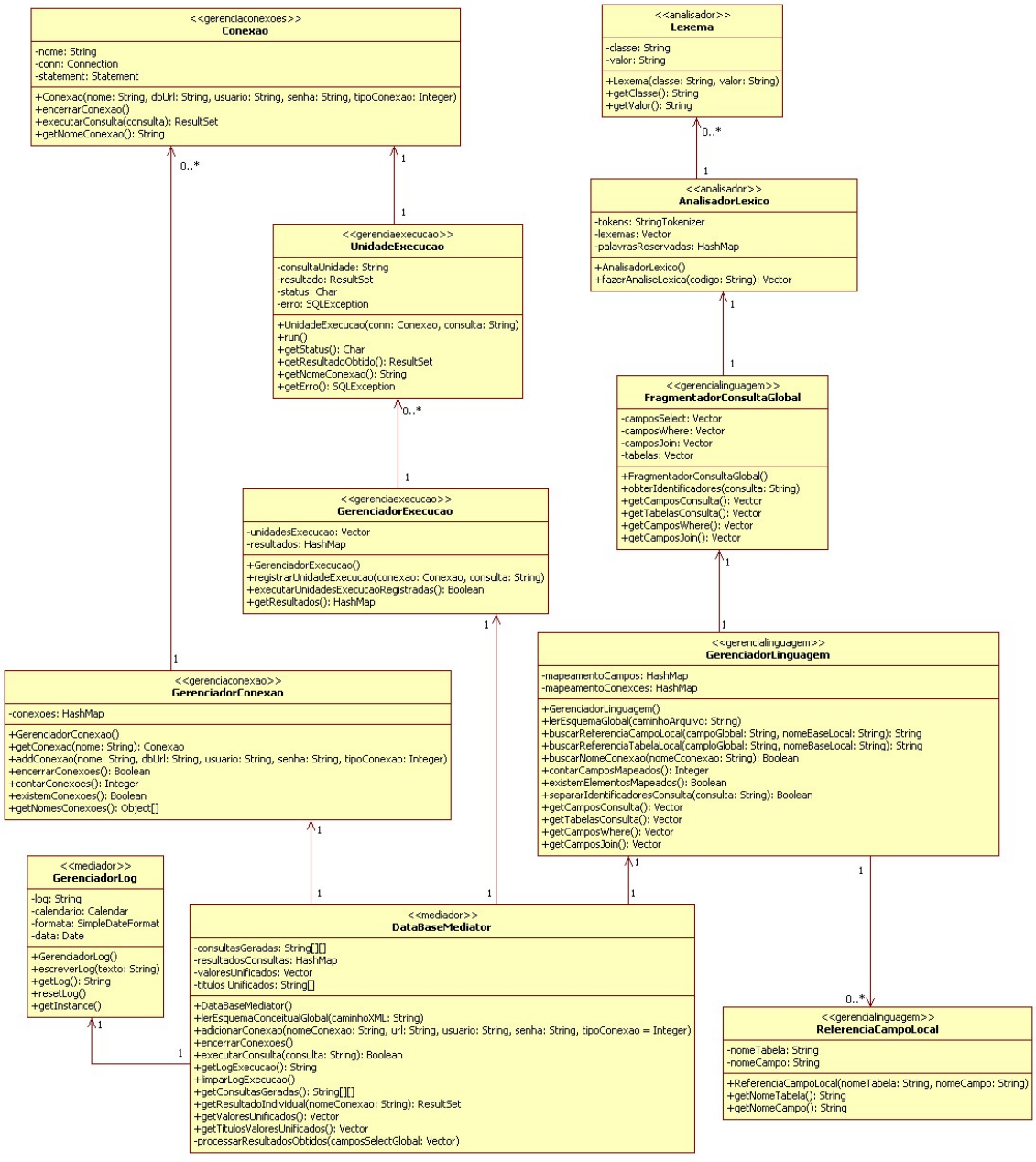

Figura 6.2: Classes do protótipo de mediador desenvolvido.

Tipo de Lexema	Exemplo(s)
Palavras reservadas	SELECT ; FROM ; WHERE
Identificadores	codigo_pessoa ; nome_pessoa ; salario_pessoa ; funcionarios
Constantes Numéricas	0,2 ; 3 ; 5000
Constantes Literais	'PEDRO%'
Operadores	* ; LIKE ; AND ; <=

Tabela 6.1: Exemplos de lexemas identificados em uma consulta definida conforme o padrão SQL-99.

léxica consiste na leitura de um código-fonte, caractere a caractere, traduzindo-o para uma seqüência de símbolos léxicos, também chamados de *tokens* ou lexemas. Na abordagem aqui tratada, o código-fonte é a consulta global que é enviada ao mediador.

Os lexemas identificados nos códigos-fonte analisados são as palavras reservadas, o identificadores, as constantes numéricas ou literais e os operadores admitidos pela linguagem. Durante o processo de análise léxica, os caracteres não significativos como espaços, por exemplo, são desconsiderados (ALENCAR PRICE; TOSCANI, 2001).

Tomando-se como exemplo a consulta a seguir, escrita em SQL-99, na tabela 6.1 pode-se observar os lexemas identificados e classificados conforme o tipo dos mesmos.

```
SELECT codigo_pessoa, nome_pessoa, salario_pessoa * 0,2
FROM funcionarios
WHERE nome_pessoa LIKE 'PEDRO%'
AND salario_pessoa * 3 <= 5000
```

O processo de análise léxica é realizado separando-se todos os *tokens* da consulta global. Para realizar essa separação foi utilizado um objeto *StringTokenizer* que possibilita a separação das partes de uma *String* (Sun Microsystems, 2008). Para cada *token* obtido é realizada a classificação do mesmo conforme os tipos definidos por (ALENCAR PRICE; TOSCANI, 2001) e criada uma nova instância da classe Lexema contendo o valor e o tipo ou classe do *token*. Cada novo Lexema identificado e classificado é armazenado em um vetor que é utilizado no processamento da consulta.

Para a facilitar a identificação das palavras reservadas e operadores admitidos pelo padrão SQL-99, foi utilizada uma estrutura auxiliar interna na classe AnalisadorLexico. Essa estrutura, um *HashMap* chamado *palavrasReservadas*, pode ser visualizada na classe em questão no diagrama apresentado na figura 6.2. A escolha da classe *HashMap* para armazenar esse tipo de estrutura deu-se pelo fato da

mesma armazenar outros objetos e buscá-los de forma bastante ágil e fácil (Sun Microsystems, 2008).

6.2 Pacote gerenciaconexoes

Neste pacote estão agrupadas as classes responsáveis por realizar o controle das conexões que o mediador possui. As classes pertencentes a esse pacote são *GerenciadorConexao* e *Conexao.*

A classe Conexão é responsável por armazenar as informações de uma conexão a um dos BDs locais que o mediador acessa. Os objetos Conexão possuem um nome, uma conexão a um banco de dados, um *Statement* e um *wrapper*. O nome da conexão serve para identificar a Conexão em questão. A conexão ao banco de dados é uma instância da *interface Connection*, retornado no momento que o *driver* JDBC realiza a conexão com o banco de dados (Sun Microsystems, 2006). O *statement* é um objeto obtido a partir da conexão e utilizado para executar comandos SQL em um banco de dados relacional (Sun Microsystems, 2008).

O *wrapper* é o objeto responsável pelo último nível de conversão de uma consulta. Ao chegar no objeto conexão, a subconsulta necessita de um último nível de conversão. Neste ponto é necessário que a subconsulta, ainda definida na linguagem global, o SQL-99, seja transcrita para a linguagem SQL compreendida pelo SGBD local ao qual a Conexão se refere. Mais detalhes sobre os *wrappers* são fornecidos ao longo deste Capítulo.

Dentre as operações disponíveis em um objeto Conexão, as relativas ao BD local são a criação do mesmo, já abrindo a conexão, o encerramento da conexão e a execução de uma consulta. No momento da criação de uma nova Conexão é necessário que os elementos básicos da conexão, seguindo o padrão (Sun Microsystems, 2006), sejam informados: tipo do SGBD, endereço do servidor, nome do BD local, usuário e senha.

A classe GerenciadorConexao é responsável pelo gerenciamento, ou seja, criação e encerramento de todas as Conexões que o mediador necessita para obter as informações. Nessa classe também se faz presente uma estrutura *HashMap* para o armazenamento de todas as Conexões que serão utilizadas pelo mediador. A utilização de um *HashMap* permite que, para cada Conexão, utilize-se o nome da mesma como chave, facilitando, dessa forma, a busca por uma determinada Conexão.

A figura 6.3 representa a estrutura de armazenamento das Conexões no GerenciadorConexão. É importante ressaltar que há somente uma Conexão atribuída a cada nome de Conexão. Além disso, nomes de Conexões repetidos não são aceitos, causando uma exceção que deverá ser tratada.

As operações fornecidas pelo GerenciadorConexao são a busca por uma Conexão,

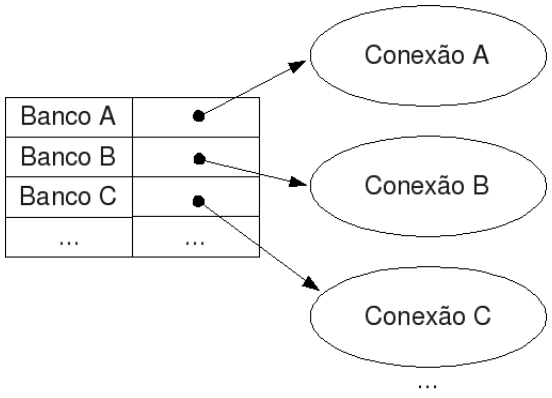

Figura 6.3: Representação da estrutura para armazenamento das Conexões.

tendo como base o nome da mesma; a criação de uma nova Conexão, informando-se todos os dados necessários; encerramento de todas as conexões; verificação da existência e contagem da quantidade de Conexões adicionadas ao GerenciadorConexao e a obtenção dos nomes de todas as Conexões existentes.

6.3 Pacote gerenciaexecucao

Este pacote agrupa as classes responsáveis por realizar o controle da execução das consultas nos Bancos de Dados locais. As classes pertencentes a esse pacote são *GerenciadorExecucao* e *UnidadeExecucao*. Essas classes podem ser visualizadas na figura 6.2.

Cada execução de uma consulta em um determinado BD local é tratado, no nível de gerenciamento de execução, como uma unidade de execução. Cada unidade de execução é uma *Thread* responsável por realizar a consulta ao seu respectivo BD local. A classe UnidadeExecução prevê algumas estruturas internas necessárias ao seu funcionamento: uma instância da classe Conexao utilizada para a execução da consulta no SGBD local, a consulta que será executada, um *ResultSet* para armazenar temporariamente os dados obtidos pela Conexao, um controle de estado, e um controle de possíveis erros que venham a ocorrer na execução da consulta.

O controle de estado da classe UnidadeExecucao admite três possíveis estados: 'X' (em execução), 'C' (concluído com sucesso) e 'E' (erro na execução). No caso de ocorrer algum erro na execução da consulta, o controle de erro recebe o *SQLException* retornado pela execução da consulta para posterior tratamento.

A classe UnidadeExecucao prevê mecanismos para obtenção de seu estado atual de execução, dos resultados obtidos com a execução da consulta e do erro ocorrido, quando for este o caso.

O GerenciadorExecucao é responsável pela criação das unidades de execução e pelo controle das mesmas. Internamente, o gerenciador de execução é formado por

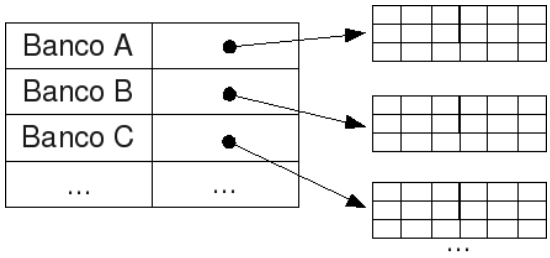

Figura 6.4: Representação da estrutura para armazenamento dos resultados obtidos pelas unidades de execução.

duas estrutura principais, um vetor onde são armazenadas as unidades de execução criadas e um *HashMap* que armazena temporariamente os resultados obtidos pelas unidades de execução. Cada resultado armazenado nesse *HashMap* tem como chave de acesso o nome da conexão da qual foi obtido.

São fornecidas pelo gerenciador de execução as funcionalidades de registro das unidades de execução bem como a execução das mesmas, além da obtenção dos resultados fornecidos pelos BDs locais.

Para efetuar o registro de uma unidade de execução faz-se necessário informar a Conexão que a mesma utilizará e a consulta a ser executada. O controle da execução inicia a execução de cada uma das unidades. Caso ocorra algum erro durante a execução de qualquer uma das unidades, o gerenciador de execução irá parar a execução de todas as demais, retornando para o mediador a exceção ocorrida. Caso a execução de todas as unidade ocorra com sucesso, os resultados obtidos são colocados no *HashMap* representado na figura 6.4, as unidades de execução atuais são destruídas e o valor *TRUE* será retornado para o mediador.

6.4 Pacote gerencialinguagem

As classes responsáveis por separar os elementos da consulta global e pela leitura do ECG, relacionando os mapeamentos entre os elementos desse e dos BDs locais estão agrupadas neste pacote. Essas classes são *GerenciadorLinguagem*, *FragmentadorConsultaGlobal* e *ReferenciaCampoLocal* e podem ser visualizadas no diagrama de classes apresentado na figura 6.2.

O gerenciador de linguagem é responsável pela leitura do ECG de um arquivo XML informado e por realizar a busca das referências locais com base no ECG lido. Além disso, o gerenciador de linguagem realiza a separação dos identificadores da consulta global utilizando para isso o FragmentadorConsultaGlobal. Para realizar suas tarefas, o gerenciador de linguagem utiliza dois *HashMaps* que armazenam, respectivamente, os nomes das conexões existentes no ECG e o mapeamento lido do ECG.

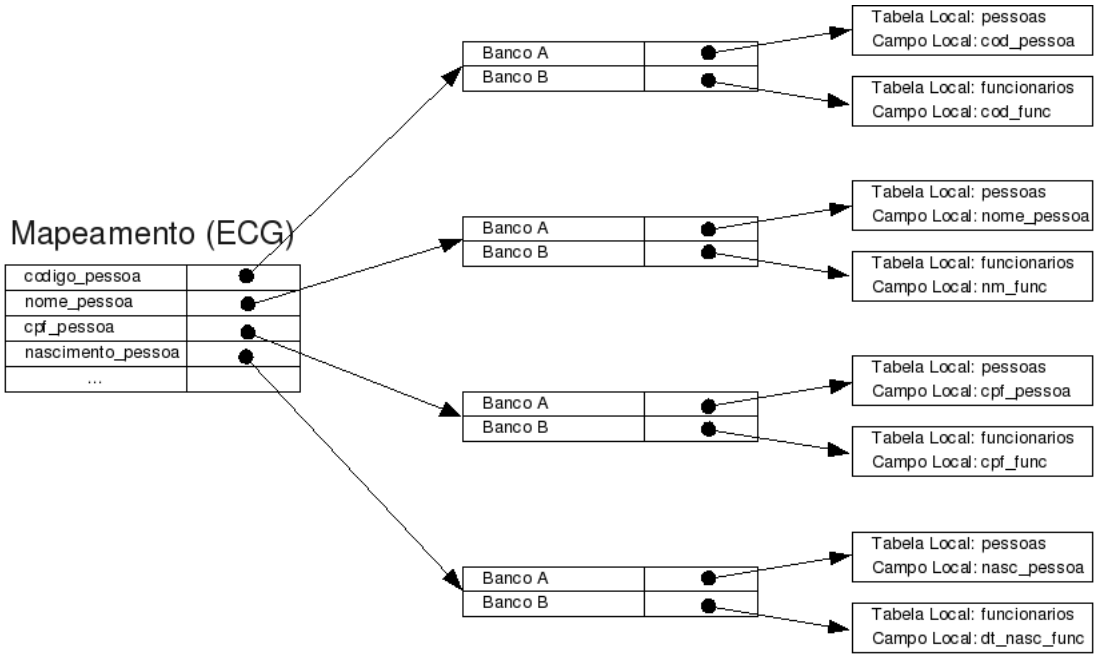

Figura 6.5: Representação da estrutura para armazenamento do mapeamento do ECG.

Ao realizar a leitura do ECG a partir do XML informado, o gerenciador de linguagem armazena o mapeamento em uma estrutura de *HashMaps* encadeados em dois níveis. Este encadeamento pode ser visualizado na figura 6.5. No primeiro nível está o mapeamento dos campos globais. Para cada campo global, há um novo mapeamento considerando as conexões apresentadas no ECG. A cada uma dessas conexões, está relacionado o objeto ReferenciaCampoLocal, que possui em sua estrutura interna o nome da tabela e do campo local.

Para realizar a busca de uma determinada tabela ou campo em um BD local, é necessário informar para o gerenciador de linguagem o nome do campo global do ECG e o nome da conexão na qual deseja-se efetuar a busca. A busca ocorre no primeiro nível do mapeamento usando-se como chave o nome do campo global. Essa busca resulta em um *HashMap* onde é realizada a busca no segundo nível, utilizando-se o nome da conexão informado. Essa busca de segundo nível obtém um objeto do tipo ReferenciaCampoLocal, onde estão armazenados os nomes da tabela e do campo local.

O fragmentador da consulta global realiza a identificação e separação dos identificadores utilizados na consulta. Ele realiza a separação dos identificadores considerando o tipo e a localização destes na consulta. A separação dada pelo tipo dos identificadores gera dois conjuntos de valores, os nomes de tabelas e o nomes de colunas da consulta. Estes últimos ainda são separados conforme a sua localização na consulta: na <*lista_select*>, na <*clausula_join*> e na <*clausula_where*>. As listas

Tipo de identificador	Identificador(es)
Tabelas	funcionarios ; enderecos
Colunas do Select	codigo_pessoa ; nome_pessoa ; salario_pessoa
Colunas do Join	codigo_endereco_funcionario ; codigo_endereco
Colunas do Where	nome_pessoa ; salario_pessoa

Tabela 6.2: Elementos identificados na fragmentação da consulta global.

separadas são utilizadas no processamento da consulta global efetuado no mediador.

Tomando-se por exemplo a consulta abaixo, os elementos retornados pela fragmentação da consulta global são apresentados na tabela 6.2.

```
SELECT codigo_pessoa, nome_pessoa, salario_pessoa * 0,2
FROM funcionarios
JOIN enderecos ON codigo_endereco_funcionario = codigo_endereco
WHERE nome_pessoa LIKE 'PEDRO%'
AND salario_pessoa * 3 <= 5000
```

O pacote de gerenciamento de linguagem ainda prevê algumas operações de controle utilizadas no mediador como, por exemplo, contagem dos campos globais mapeados e verificação da existência de campos globais no arquivo XML.

6.5 Pacote mediador

Este pacote agrupa as classes *DataBaseMediator* e *GerenciadorLog*. Estas classes são responsáveis pelas atividades do processamento de consulta apresentadas no capítulo 3. Todos os gerenciadores apresentados até o momento, suas estruturas auxiliares e operações são acessadas através de uma instância da classe DataBaseMediator, ou seja, para utilizar o protótipo de mediador aqui apresentado, é necessário criar somente uma instância do mesmo.

A classe GerenciadorLog foi adicionada ao pacote como uma funcionalidade extra, sendo responsável por escrever um *log* de todas as operações que o mediador executa. A escrita deste *log* ocorre de forma automática, ficando a cargo do desenvolvedor da aplicação que utiliza o mediador, realizar a leitura ou a gravação do mesmo em um arquivo no disco, caso seja conveniente.

A cargo da classe DataBaseMediator estão as tarefas de preparação do ambiente e de processamento da consula global. A preparação do ambiente consiste na leitura do ECG a partir do arquivo XML, criação e encerramento das conexões aos BDs locais e fornecimento do log de execução para a aplicação. As tarefas ligadas ao

processamento da consulta global consistem na execução da consulta global e fornecimento do resultado obtido após a execução das subconsultas nos BDs locais. Para fins acadêmicos, que podem ser verificados na aplicação de simulação e execução de consultas, foram adicionados mecanismos de obtenção das subconsultas geradas pelo mediador e dos resultados obtidos em cada um dos BDs locais.

Para realizar as atividades de sua responsabilidade a classe DataBaseMediator conta com as seguintes estruturas internas: um gerenciador de conexões, um gerenciador de linguagem, um gerenciador de *log* e um gerenciador de execução. As funcionalidades e competências de cada uma dessas estruturas já foram explanadas ao longo deste capítulo. Além destas estruturas, fazem-se presentes estruturas que armazenam as subconsultas geradas, os resultados individuais obtidos em cada BD local e o resultado após a união dos resultados individuais.

6.5.1 Processamento da consulta global no protótipo de mediador desenvolvido

O processamento da consulta global é realizado baseando-se no algoritmo do projeto *J-Integrator* (HARA et al., 2006), apresentado no capítulo 3. Este algoritmo baseia-se na idéia da obtenção dos atributos desejados em todas as tabelas envolvidas na consulta global, realizando consultas aos BDs locais e armazenando os resultados em um BD local, aqui chamado de BD resultante. O mediador deverá possuir acesso e privilégios para efetuar a criação de tabelas e inserção de dados neste BD resultante. Para cada tabela local identificada na consulta global, uma tabela é criada no BD resultante e os dados obtidos são armazenados na mesma. Após a obtenção dos dados necessários, uma nova consulta é feita sobre os dados do BD resultante para então, retornar à aplicação um único conjunto de dados. Esta abordagem mostra-se viável para as situações em que o volume de dados seja elevado. O algoritmo detalhado do projeto *J-Integrator* pode ser encontrado no Anexo A.

O processamento da consulta global no protótipo de mediador considera os passos do processamento de consulta apresentados no capítulo 04 e complementados no capítulo 05 com o serviço de junção dos dados para devolução à aplicação.

A seguir pode-se visualizar o algoritmo utilizado no processamento da consulta global.

```
1 Entrada: consulta_global, ecg, conexoes
2 Saída: dados_cg

4 atribSelect[] := colunas_select(consulta_global)
5 atribWhere[] := colunas_where(consulta_global)
6 atribJoin[] := colunas_join(consulta_global)
7 tabelas[] := tabelas(consulta_global)
```

```
para cada conexao faça
    clausula_select := separar_select(consulta_global)
    clausula_from := separar_from(consulta_global)
    clausula_join := separar_join(consulta_global)
    clausula_where := separar_where(consulta_global)

    para cada coluna_select em atribSelect[] faça
        ref_loc := buscar_ref_loc(atribSelect, ecg)
        substituir(atribSelect, ref_loc, clausula_select)
    fim para

    para cada coluna_join em atribJoin[] faça
        ref_loc := buscar_ref_loc(atribJoin, ecg)
        substituir(atribJoin, ref_loc, clausula_join)
    fim para

    para cada coluna_where em atribWhere[] faça
        ref_loc := buscar_ref_loc(atribWhere, ecg)
        substituir(atribWhere, ref_loc, clausula_Where)
    fim para

    para cada tabela em tabelas[] faça
        ref_loc := buscar_ref_loc(tabela, ecg)
        substituir(tabela, ref_loc, clausula_from)
        substituir(tabela, ref_loc, clausula_join)
    fim para

    subconsultas[] := clausula_select + clausula_from +
                      clausula_join + clausula_where
fim para

para cada subconsulta em subconsultas[] faça
    resultados_parciais[] := executar(subconsulta)
fim para

para cada resultado_parcial em resultados_parciais[] faça
    dados_cg := dados_cg + resultado_parcial
fim para
```

```
48 retorna dados_cg
```

O algoritmo ilustrado necessita que seja fornecida a consulta global e o mapeamento do ECG para sua execução. Ao término da execução, é retornado um único conjunto contendo os dados obtidos. Nas linhas 04 a 07, os elementos da consulta são obtidos conforme sua localização e tipo, ou seja, as tabelas e os nomes dos campos são separados, sendo os últimos separados conforme sua localização na consulta.

Para cada conexão registrada no mediador, o processo de identificação dos mapeamentos é realizado. Nas linhas 10 a 13 a consulta global é separada em suas principais partes: <clausula_select>, <clausula_from>, <clausula_join> e <clausula_where>. Essa divisão é realizada para manter a estrutura da consulta, ou seja, manter as restrições fornecidas e as operações que podem estar na lista de atributos da <clausula_select>.

No bloco formado pelas linhas 15 a 18, os nomes dos campos globais contidos na <clausula_select> são buscados no mapeamento considerando-se a conexão corrente e, posteriormente, substituídos na <clausula_select> pelos mapeamentos encontrados. Processo semelhante ocorre com a <clausula_join> e a <clausula_where>, respectivamente, nos blocos formados pelas linhas 20 a 23 e 25 a 28.

Nas linhas 30 a 34 ocorre a busca dos nomes das tabelas no ECG e posterior substituição pelos nomes locais encontrados. Essa substituição ocorre na <clausula_from> e na <clausula_join>, locais onde há a referência a tabelas globais.

Neste momento da execução, a consulta já está definida nos termos do BD local corrente. Basta então realizar a junção das cláusulas que foram anteriormente separadas e armazenar a consulta (linhas 36 e 37).

O bloco formado pelas linhas 40 a 42 é responsável pela execução das subconsultas geradas. Após a execução das mesmas, é realizada a união dos resultados obtidos (linhas 44 a 46).

A figura 6.6 mostra o diagrama da seqüência dos passos da execução de uma consulta nos BDs locais enquanto a figura 6.7 exibe o diagrama de atividades da execução de uma consulta global.

A aplicação do usuário deverá fazer com que o mediador realize a leitura do ECG e a inclusão das conexões aos BDs locais antes de processar uma consulta global. Essas atividades fazem parte da preparação do ambiente para execução de consultas. Nesta fase são feitas algumas consistências que merecem destaque. Ao adicionar uma nova conexão, o nome dado para a mesma deverá estar contido como uma fonte de informação no ECG. Conseqüentemente, somente é possível adicionar uma conexão a um BD local se a leitura do ECG já tiver ocorrido. Caso alguma destas restrições tenha sido violada, não haverá processamento da consulta global,

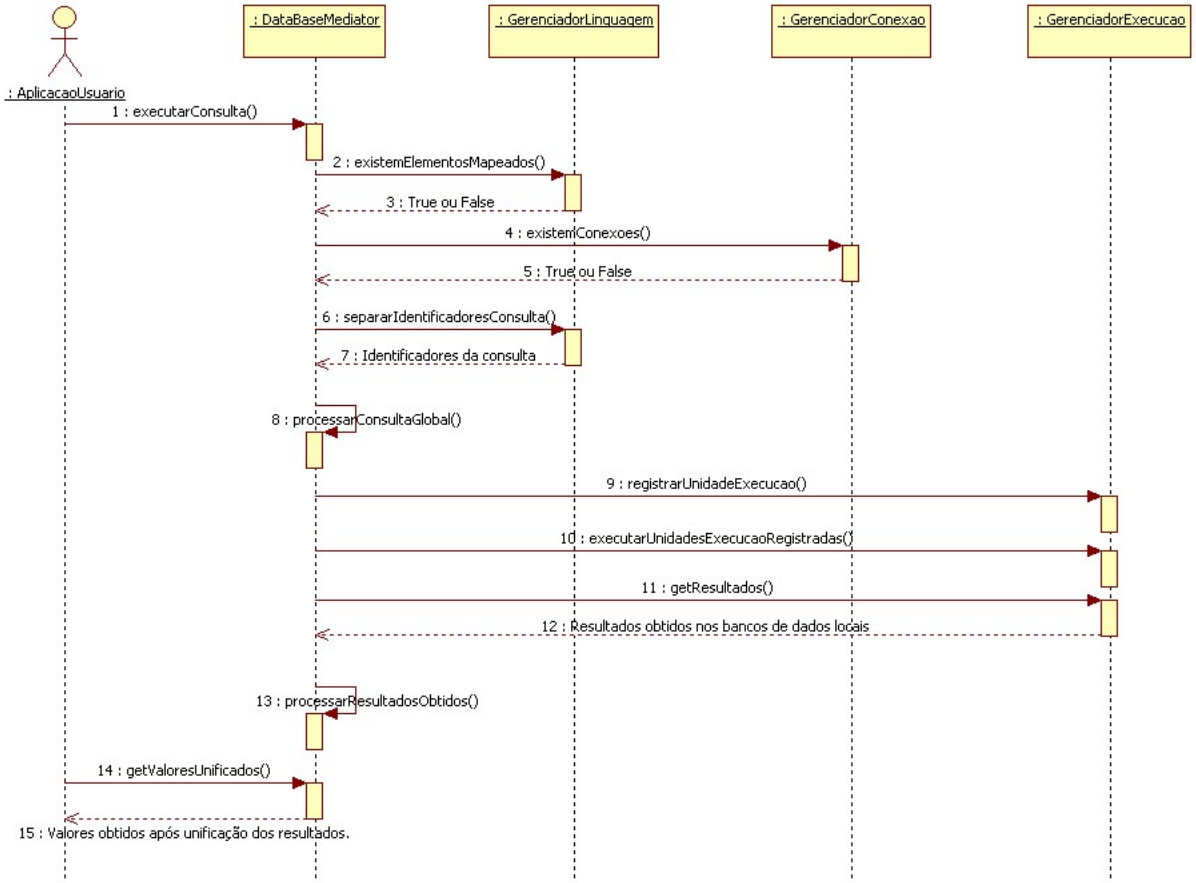

Figura 6.6: Diagrama de Seqüência da execução da consulta global no protótipo de mediador.

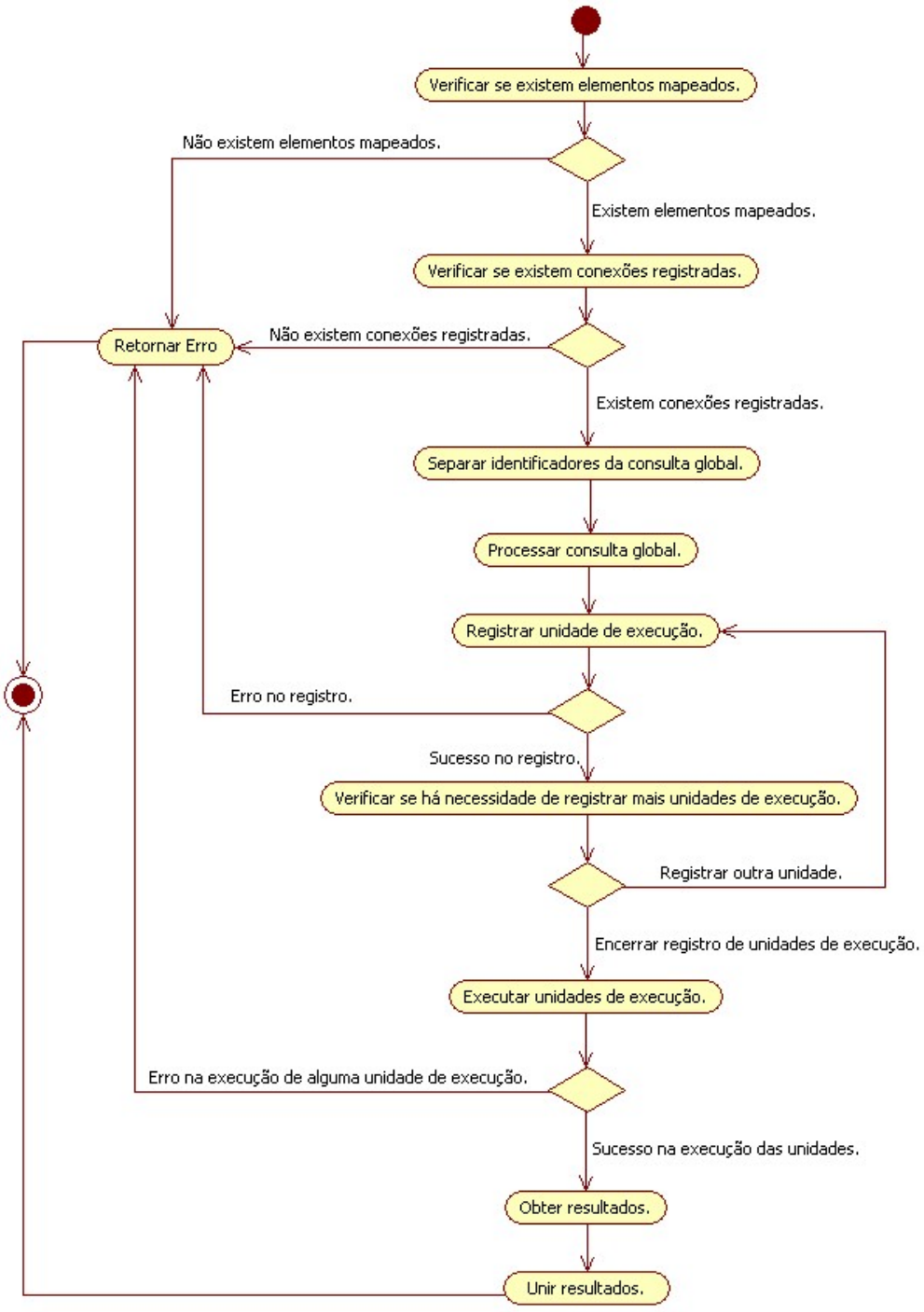

Figura 6.7: Diagrama de Atividades da execução da consulta global no protótipo de mediador.

Figura 6.8: Diagrama de atividades da preparação do ambiente para a execução de consultas globais.

sendo retornada para a aplicação do usuário a exceção correspondente.

Na figura 6.8 é apresentado o diagrama de atividades que exemplifica as restrições impostas e os comportamentos esperados do protótipo de mediador na preparação do ambiente para o processamento das consultas globais.

6.6 Conversores

Juntamente com o protótipo de mediador foram desenvolvidos dois conversores ou *wrappers* para os SGBDs MySQL e PostgreSQL. Tais elementos têm a responsabilidade de converter a linguagem da consulta global na linguagem do seu respectivo SGBD. Além disso, é função do *wrapper* realizar a conversão dos tipos de dados do SGBD para um tipo de dados canônico, cujo mediador seja capaz de manipular (GARCIA-MOLINA; ULMAN; WIDOM, 2001).

A tarefa de realizar a conversão dos tipos de dados dos SGBDs para um tipo de dados canônico não se faz necessária na abordagem utilizada, pois a conexão com os BDs locais é realizada utilizando-se interfaces JDBC específicas para cada SGBD, cuja padronização prevê quais tipos de dados devem ser retornados na execução de uma consulta (Sun Microsystems, 2006). Dessa forma é possível utilizar conjuntamente com o mediador qualquer *driver* JDBC que obedeça o padrão especificado sem haver a necessidade da conversão dos tipos de dados obtidos das fontes.

Quanto à tarefa de conversão da linguagem de consulta global, fez-se necessário

o desenvolvimento de mecanismos de conversão somente no *wrapper* para o SGBD MySQL. Isto deu-se pelo fato de que o SGBD PostgreSQL é implementado dentro do padrão SQL-99, a linguagem de consulta global utilizada pelo mediador (PostgreSQL Br, 2009).

Já o SGBD MySQL implementa a maior parte das funcionalidade previstas no padrão SQL-99. Dentre as opções do padrão SQL-99 aceitos pelo mediador, somente as operações de concatenação e *FULL JOIN* não são implementadas conforme o padrão SQL-99 (MySQL, 2008). Neste caso, fez-se necessária a implementação de mecanismos de conversão no *wrapper* MySQL para que a subconsulta atenda às especificações do SGBD em questão.

De acordo com o padrão SQL-99, o operador de concatenação é o caracter '|' duplicado, ou seja, '||'. Entretanto, o SGBD MySQL implementa a operação de concatenação através de uma função chamada *concat*. Essa função é chamada passando-se como parâmetros os elementos que deverão ser concatenados. O mecanismo de conversão do *wrapper* MySQL identifica todos os elementos que necessitam ser concatenados e estão separados por '||' na consulta SQL-99, transformando-os em uma chamada da referida função.

A outra operação prevista no padrão SQL-99 e não implementada no MySQL é o tipo de junção conhecido por *FULL JOIN* (MySQL, 2008). Segundo (SILBERSCHATZ; KORTH; SUDARSHAN, 1999), a junção total ou *FULL JOIN* é equivalente à união de uma junção à esquerda (*LEFT JOIN*) com uma junção à direita (*RIGHT JOIN*). É baseado nesta idéia que o *wrapper* MySQL realiza a conversão da consulta SQL-99 quando houver a ocorrência de uma junção total.

O anexo B relaciona as opções do SQL-99 aceitas pelo mediador com suas respectivas implementações nos SGBDs MySQL e PostgreSQL.

7 APLICAÇÃO PARA SIMULAÇÃO DE USO DO PROTÓTIPO DE MEDIADOR

Juntamente com o protótipo do mediador foi desenvolvida uma aplicação para execução de consultas sobre diversos SGBDs locais utilizando o prototipo de mediador. Essa aplicação de simulação é composta de três partes: uma janela para digitação da consulta global e visualização do conjunto de resultados obtidos, uma janela para acompanhamento do log de execução da consulta e uma janela para verificação das subconsultas geradas e dos resultados individuais obtidos em cada SGBD local. Essa última, é instanciada uma vez para cada conexão aberta pelo simulador.

A figura 7.1 ilustra a tela principal do simulador, onde a consulta global é digitada e o conjunto de dados obtidos é exibido. Nessa tela pode-se observar o campo para digitação de uma consulta global, a área onde os resultados obtidos são exibidos e um indicador de estado, ou seja, se a consulta global está executando, se foi executada com sucesso ou se houve erro na execução. Existem ainda os seguintes botões de comando: executar, utilizado para iniciar a execução de uma consulta global e limpar, responsável por limpar as informações exibidas.

A figura 7.2 mostra a janela de acompanhamento do log de execução. Nesta janela existe somente a área onde o texto do log é apresentado. Cada evento do log fica associado à hora de sua ocorrência.

As figuras 7.3 e 7.4 mostram as janelas de acompanhamento das subconsultas para os SGBDs PostgreSQL e MySQL, respectivamente. Em cada uma dessas janelas pode-se observar uma área onde é mostrada a subconsulta gerada pelo mediador para o SGBD ao qual a sua respectiva conexão está ligada. Além disso, é possível observar, ao término da execução, os resultados obtidos na sua respectiva fonte de dados. Da mesma forma que na janela principal do simulador, há também uma área onde é mostrado o estado atual da execução da subconsulta no SGBD local.

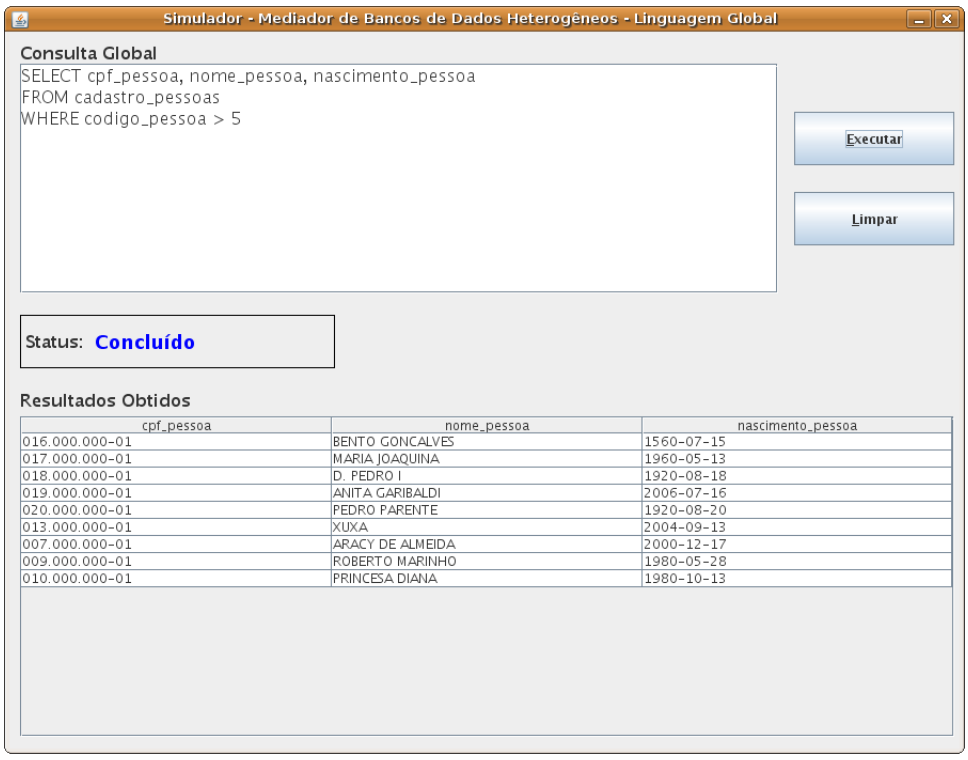

Figura 7.1: Janela principal do simulador após a execução de uma consulta global.

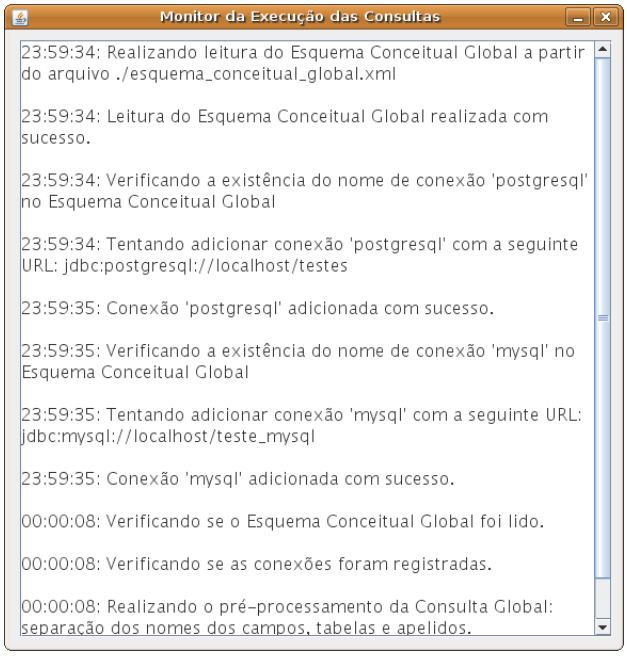

Figura 7.2: Janela de acompanhamento do log de execução.

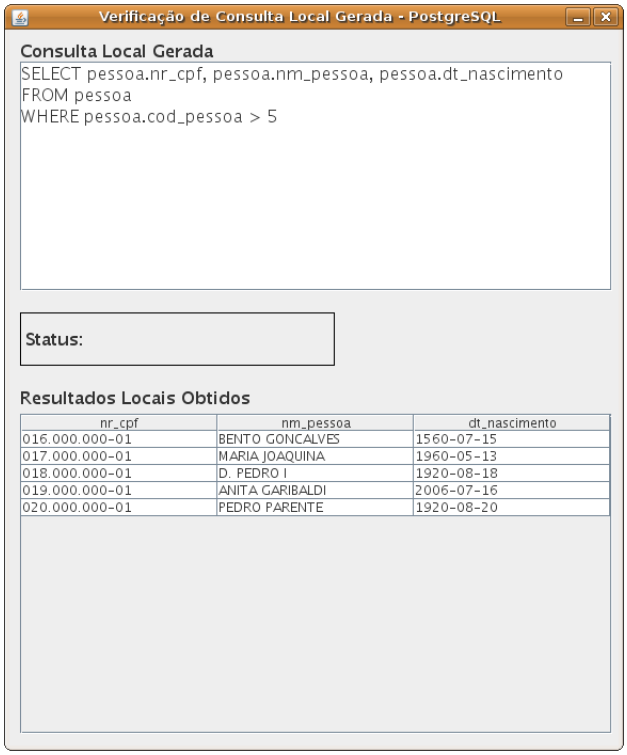

Figura 7.3: Janela de acompanhamento da execução da subconsulta no PostgreSQL.

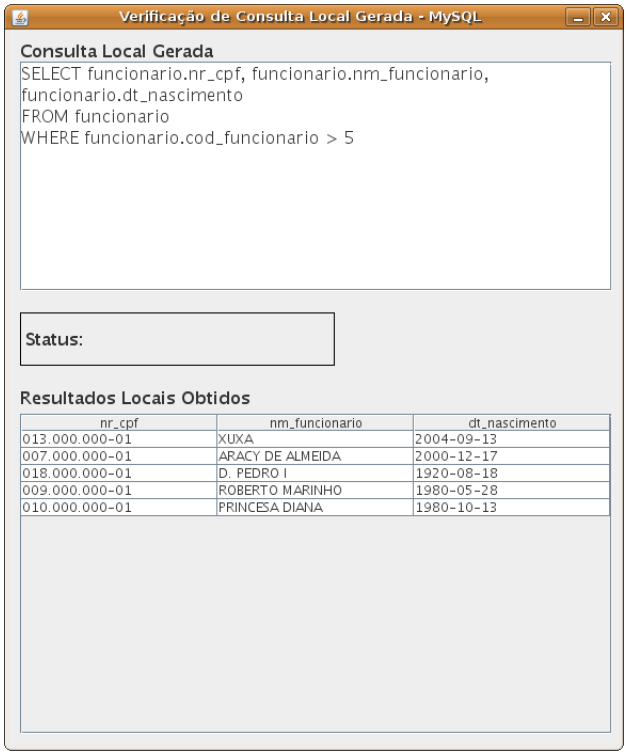

Figura 7.4: Janela de acompanhamento da execução da subconsulta no MySQL.

7.1 Testes realizados

Os testes relatados nesta seção utilizaram bases de dados públicas disponibilizadas pelo governo dos Estados Unidos. Estas bases contêm informações de estimativas populacionais baseadas nos dados dos censos realizados nos Estados Unidos desde o ano 2000. As consultas e resultados apresentados nos testes estão baseados nos modelos de dados apresentados no Anexo C, cujo mapeamento global (ECG) pode ser visualizado no Anexo D.

Para cada teste apresentado, será mostrada a consulta global, as subconsultas geradas, os resultados obtidos em cada fonte de dados e a união destes resultados.

7.1.1 Execução de Consulta Global Simples

Neste teste, uma consulta simples foi realizada utilizando o protótipo de mediador. O objetivo da consulta é recuperar os códigos e nomes das localizações cujos códigos estejam entre 1025 e 1045.

A consulta global a seguir foi informada para a aplicação de simulação.

```
SELECT codigo_localizacao, nome_localizacao
FROM localizacoes
WHERE codigo_localizacao BETWEEN 1025 AND 1045;
```

Para o BD armazenado no PostgreSQL, a subconsulta abaixo foi gerada. A tabela 7.1 mostra os resultados obtidos por esta subconsulta.

```
SELECT localizacao.cod_localizacao, localizacao.nm_localizacao
FROM localizacao
WHERE localizacao.cod_localizacao BETWEEN 1025 AND 1045;
```

Já para o BD armazenado no MySQL, a subconsulta a seguir foi gerada e, na tabela 7.2, são mostrados os resultados obtidos por esta subconsulta.

```
SELECT localizacao.cod_localizacao, localizacao.nm_localizacao
FROM localizacao
WHERE localizacao.cod_localizacao BETWEEN 1025 AND 1045;
```

É importante ressaltar que as subconsultas geradas para ambos BDs locais são idênticas pelo fato dos atributos locais de ambas as bases de dados possuírem os mesmos nomes. Coincidentemente, os dados recuperados também são os mesmos, uma vez que nas tabelas de onde os dados foram recuperados estão armazenados os dados das cidades norte-americanas. A tabela 7.3 ilustra os dados obtidos após o processo de união dos conjuntos de dados locais.

cod_localizacao	nm_localizacao
1025	Clarke County, Alabama
1027	Clay County, Alabama
1029	Cleburne County, Alabama
1031	Coffee County, Alabama
1033	Colbert County, Alabama
1035	Conecuh County, Alabama
1037	Coosa County, Alabama
1039	Covington County, Alabama
1041	Crenshaw County, Alabama
1043	Cullman County, Alabama
1045	Dale County, Alabama

Tabela 7.1: Resultados obtidos na execução da subconsulta enviada ao SGBD PostgreSQL - Consulta Global Simples

cod_localizacao	nm_localizacao
1025	Clarke County, Alabama
1027	Clay County, Alabama
1029	Cleburne County, Alabama
1031	Coffee County, Alabama
1033	Colbert County, Alabama
1035	Conecuh County, Alabama
1037	Coosa County, Alabama
1039	Covington County, Alabama
1041	Crenshaw County, Alabama
1043	Cullman County, Alabama
1045	Dale County, Alabama

Tabela 7.2: Resultados obtidos na execução da subconsulta enviada ao SGBD MySQL - Consulta Global Simples

codigo_localizacao	nome_localizacao
1025	Clarke County, Alabama
1027	Clay County, Alabama
1029	Cleburne County, Alabama
1031	Coffee County, Alabama
1033	Colbert County, Alabama
1035	Conecuh County, Alabama
1037	Coosa County, Alabama
1039	Covington County, Alabama
1041	Crenshaw County, Alabama
1043	Cullman County, Alabama
1045	Dale County, Alabama

Tabela 7.3: Resultados obtidos na execução da consulta global após o processo de união - Consulta Global Simpes.

7.1.2 Execução de Consulta Global com Junções

Neste teste, foi executada uma consulta envolvendo junções entre tabelas que estão na mesma base de dados. Todas as tabelas envolvidas possuem mapeamento no ECG. A consulta global executada pode ser visualizada a seguir. Nesta consulta, a intenção é obter o nome da localização, a estimativa do total da população no ano de 2007 e as estimativas das quantidades de homens e mulheres abaixo dos 20 anos de idade no mesmo ano para as cidades cujo nome se inicia por *Newton*.

```
SELECT nome_localizacao, valor_populacao_2007,
       homem_menor_20 + homem_menor_10,
       mulher_menor_20 + mulher_menor_10
FROM localizacoes
INNER JOIN dados_populacao_geral ON
           codigo_localizacao = codigo_local_populacao
INNER JOIN dados_homens ON
           codigo_localizacao = codigo_local_homem
INNER JOIN dados_mulheres ON
           codigo_localizacao = codigo_local_mulher
WHERE nm_localizacao like 'Newton%';
```

Após o processamento da consulta global, a subconsulta a seguir foi gerada para o SGBD PostgreSQL. Na tabela 7.4, os resultados obtidos na execução dessa subconsulta são mostrados.

```
SELECT localizacao.nm_localizacao, populacao.pop_2007,
       homem.menos_20 + homem.menos_10,
```

nm_localizacao	populacao_2007	homens_20	mulheres_20
Newton County, Arkansas	8335	980	869
Newton County, Georgia	95723	16258	15536
Newton County, Indiana	14004	1686	1632
Newton County, Mississippi	22334	3457	3284
Newton County, Missouri	55994	7770	7504
Newton County, Texas	13782	1720	1559

Tabela 7.4: Resultados obtidos na execução da subconsulta enviada ao SGBD PostgreSQL - Consulta Global com Junções.

```
       mulher.menos_20 + mulher.menos_10
FROM localizacao
INNER JOIN populacao ON
           localizacao.cod_localizacao = populacao.cod_localizacao
INNER JOIN homem ON
           localizacao.cod_localizacao = homem.cod_localizacao
INNER JOIN mulher ON
           localizacao.cod_localizacao = mulher.cod_localizacao
WHERE nm_localizacao like 'Newton%';
```

A subconsulta gerada para o SGBD MySQL é mostrada a seguir. Os resultados obtidos por esta subconsulta podem ser visualizados na tabela 7.5

```
SELECT localizacao.nm_localizacao, populacao.vlr_populacao_2007,
       homem.vlr_menor_20 + homem.vlr_menor_10,
       mulher.vlr_menor_20 + mulher.vlr_menor_10
FROM localizacao
INNER JOIN populacao ON
           localizacao.cod_localizacao = populacao.cod_localizacao
INNER JOIN homem ON l
           ocalizacao.cod_localizacao = homem.cod_localizacao
INNER JOIN mulher ON
           localizacao.cod_localizacao = mulher.cod_localizacao
WHERE nm_localizacao like 'Newton%';
```

Na tabela 7.6 a união dos resultados obtidos pelas subconsultas é mostrada. Este novo conjunto de dados foi retornado para a aplicação de simulação após a execução das consultas globais.

nm_localizacao	populacao_2007	homens_20	mulheres_20
Newton County, Arkansas	8339	1008	888
Newton County, Georgia	96019	16017	15283
Newton County, Indiana	14014	1742	1660
Newton County, Mississippi	22329	3464	3312
Newton County, Missouri	56038	7668	7481
Newton County, Texas	13827	1772	1605

Tabela 7.5: Resultados obtidos na execução da subconsulta enviada ao SGBD MySQL - Consulta Global com Junções.

nm_localizacao	populacao_2007	homens_20	mulheres_20
Newton County, Arkansas	8335	980	869
Newton County, Arkansas	8339	1008	888
Newton County, Georgia	95723	16258	15536
Newton County, Georgia	96019	16017	15283
Newton County, Indiana	14004	1686	1632
Newton County, Indiana	14014	1742	1660
Newton County, Mississippi	22334	3457	3284
Newton County, Mississippi	22329	3464	3312
Newton County, Missouri	55994	7770	7504
Newton County, Missouri	56038	7668	7481
Newton County, Texas	13782	1720	1559
Newton County, Texas	13827	1772	1605

Tabela 7.6: Resultados obtidos na execução da consulta global após o processo de união - Consulta Global com Junções.

7.1.3 Erro na Execução de Consulta Global

Este teste tem por objetivo demonstrar o cancelamento da execução de uma consulta global caso algum erro seja detectado ou algum dos SGBDs locais não esteja disponível. Para realizar este teste, será tomado como exemplo de cosulta global a consulta apresentada na subseção 7.1.2.

Para esta simulação o SGBD PostgreSQL teve sua execução interrompida após ter sido efetuada a conexão com o mesmo. No momento da execução da consulta global, o seguinte erro foi retornado.

"org.postgresql.util.PSQLException: FATAL: terminando conexão por causa de um comando do administrador."

Outra situação que pode ocorrer é a má formulação de consultas globais. Neste caso, as subconsultas geradas também estarão sujeitas aos mesmos erros apresentados na consulta global, podendo ocasioar um erro de execução e, conseqüentemente, o cancelamento da execução da transação global.

Para simular esta situação, a consulta a seguir será utilizada.

```
SELECT codigo_localizacao, nome_localizacao
FROM localizacoes
WHERE codigo_localizacao << 8;
```

Na consulta apresentada, existe um erro no operador relacional utilizado. A mensagem retornada pelo mediador, informando o erro é apresentada a seguir.

"org.postgresql.util.PSQLException: ERRO: argumento do WHERE deve ser do tipo boolean, e não do tipo integer."

8 CONCLUSÃO

É crescente a necessidade das organizações de possuir suas informações integradas. Entretanto, muitas vezes essas informações são acessadas por distintas aplicações e estão armazenadas em BDs variados. Nestes casos, freqüentemente não é possível adquirir novos sistemas e SGBDs, ou então não é viável a migração dessas informações. Neste contexto, a solução que tem sido adotada em muitos casos é a manutenção dos SGBDs existentes e a construção de uma camada para realizar a integração dessas informações.

Este trabalho teve por objetivo realizar um estudo geral sobre as propostas existentes de integraçao de informações. Das propostas estudadas, a integração por mediadores foi enfatizada por não depender de cópias de informações entre BDs. Nesta abordagem é possível acessar os dados atualizados das bases de dados locais mesmo que essas ainda sejam modificadas pelas suas respectivas aplicações. Entretanto, mesmo dentro da abordagem de mediadores, foi dado maior enfoque ao processamento de consultas por existirem muitas diferenças entre as linguagens implementadas nos principais SGBDs utilizados atualmente, sendo este um dos fatores que mais dificulta a integração de informações.

Ainda como objetivo deste trabalho e, com base nos estudos realizados, foi definida uma arquitetura e realizado o desenvolvimento de um protótipo de mediador.

Com base nos estudos realizados, pode-se perceber que as propostas de mediadores são bastante parecidas, ou seja, não variam muito entre os projetos analisados. As pequenas variações identificadas são adaptações para atender a requisitos dos projetos em que estão inseridos. É difícil determinar um plano de execução de consultas globais em mediadores que atenda a todos os tipos de situação, ou então prever que essa seja a solução para a integração de qualquer ambiente heterogêneo. Entretanto, a proposta aqui apresentada procura ser o máximo possível portável, tendo como objetivo a possibilidade ser utilizada com grande parte dos SGBDs utilizados atualmente, desde que esses atendam ao padrão SQL-99.

O protótipo de mediador desenvolvido mostrou-se eficiente para a manipulação de pequenas quantidades de dados. Para a manipulação de grandes quantidades de

informações existem abordagens distintas da utilizada que, de acordo com as pesquisas realizadas, são bastante eficientes. Entretanto, há a possibilidade de realizar a expansão, com algumas adaptações, do protótipo afim de que o mesmo torne-se mais eficiente na manipulação de grandes quantidades de dados.

Como trabalhos futuros, pode-se colocar a expansão da linguagem de consulta tratada neste trabalho, atendendo a outras opções previstas no padrão SQL-99. Além disso, pode-se trabalhar na união dos dados obtidos das bases de dados locais, bem como na aplicação dos serviços oferecidos pelo mediador desenvolvido. Pode-se ainda realizar pesquisas e trabalhos na otimização das técnicas existentes para o processamento de consultas globais e junção das informações obtidas. É possível ainda estender estudos à integração de BDHs realizando relacionamentos entre elementos de distintos BDs. Pode-se também estudar técnicas de inteligência artificial para automatizar a integração de esquemas, solucionando diferenças semânticas entre os elementos dos BDs locais. Outra possível abordagem de estudo para solucionar a heterogeneidade semântica é o uso de ontologias, ou seja, um conjunto de conceitos em um determinado domínio de informações, para realizar os mapeamentos de esquemas. Enfim, são inúmeras as possibilidades de ampliação dos estudos na área, com a possibilidade da criação de técnicas cada vez mais otimizadas para prover a integração de BDHs.

REFERÊNCIAS

ABREU, B. R. C. de. **Unindo Sistemas e Bancos de Dados Distribuídos**. 2006. Monografia do Trabalho de Conclusão de Curso para obtenção do título de Bacharel em Ciência da Computação — Universidade Federal de Pernambuco, Recife, PE.

ALENCAR PRICE, A. M. de; TOSCANI, S. S. **Implementação de Linguagens de Programação**: Compiladores. Porto Alegre: Sagra Luzzatto, 2001.

ANSI-SQL. **Database Language SQL**. [S.l.]: ANSI/ISO/IEC International Standard (IS), 1999. (ISO/IEC 9075:1999).

ARAUJO DUARTE, D. E. de. **Processamento de Consultas em Bancos de Dados Heterogêneos e Distribuídos**. 2004. Monografia do Trabalho de Conclusão de Curso para obtenção do título de Bacharel em Ciência da Computação — Universidade de Caxias do Sul, Caxias do Sul, RS.

BALLARDIN, V. **Integração de Bancos de Dados Biológicos**. 2004. Projeto de Diplomação — Universidade de Caxias do Sul, Caxias do Sul, RS.

BATINI, C.; LENZERINI, M.; NAVATHE, S. B. A Comparative Analysis of Methodologies for Database Schema Integration. **ACM Computing Surveys**, New York, v.18, n.4, p.323–364, 1986.

BOUGUETTAYA, A.; BENATALLAH, B.; ELMAGARMID, A. **Interconnecting Heterogeneous Information Systems**. Boston: Kluwer, 1998.

CASANOVA, M. A. **Sistemas Gerência de Bancos de Dados Distribuídos**. Rio de Janeiro: Campus, 1985.

COSTA, T. A. **O Gerenciador de Consultas de um Sistema de Integração de Dados**. 2005. Dissertação de Mestrado em Ciência da Computação — Universidade Federal de Pernambuco, Recife, PE.

DARONCO, E. L. **Metodologias de Integração de Bancos de Dados Distribuídos**. 2000. Trabalho Individual I de Mestrado em Ciência da Computação — Universidade Federal do Rio Grande do Sul, Porto Alegre, RS.

DATE, C. J. **Introdução a Sistemas de Bancos de Dados**. Rio de Janeiro: Elsevier, 2004.

ELMAGARMID, A.; RUSINKIEWCZ, M.; SHETH, A. **Management of Heterogeneous and Autonomous Database Systems**. San Francisco: Morgan Kaufmann, 1999.

ELMASRI, R.; NAVATHE, S. B. **Sistemas de Bancos de Dados**. São Paulo: Addison-Wesley, 2005.

GARCIA-MOLINA, H.; ULMAN, J. D.; WIDOM, J. **Implementação de Sistemas de Bancos de Dados**. Rio de Janeiro: Campus, 2001.

GRANVILLE, L. Z. **Componentes de Software para Integração de Aplicações WEB com Base de Dados em Geral e Base de Dados Legados**. 1998. Trabalho Individual II da Pós-Graduação em Ciência da Computação — Universidade Federal do Rio Grande do Sul, Porto Alegre, RS.

HARA, C.; NOGUEIRA, V. C.; R. RUTHES, E. da; SCOPIM, K.; SUNYE, M. S. J-Integrator: Uma Ferramenta para Integração de Bancos de Dados Heterogêneos. **Anais do 3º CONTECSI (International Conference on Information Systems and Technology Management)**, Curitiba, PR, 2006.

KORTH, H. F. **Sistemas de Bancos de Dados**. São Paulo: Makron Books, 1995.

LIMA, F.; MELO, R. N. Consultas em Sistemas Gerenciadores de Bases de Dados Heterogêneos. , Rio de Janeiro, 1999.

LIMA, F. O. **A Sociedade Digital**: o impacto da tecnologia na sociedade, na cultura, na educação e nas organizações. Rio de Janeiro: Qualitymark, 2000.

LITWIN, W.; MARK, L.; ROUSSOPOULOS, N. Interoperability of Multiple Autonomous Databases. **ACM Computing Surveys**, New York, v.22, n.3, p.267–293, 1990.

MENG, W.; CLEMENT, Y. Query Processing in Multidatabase Systems. **ACM Computing Surveys**, New York, p.551–572, 1995.

MySQL. **Documentação do MySQL 5.4**. Disponível em: <http://dev.mysql.com/doc/>. Acesso em: novembro 2009.

MySQL. **About MySQL**. Disponível em: <http://www.mysql.com/about/>. Acesso em: maio 2009.

OLIVEIRA CUNHA, S. L. de. **Gerenciamento de Dados em Bancos de Dados Distribuídos**. 2003. Monografia apresentada ao Curso de Ciência da Computação do Centro Universitário do Triângulo - Unit, como requisito básico à obtenção do Grau de Bacharel em Ciência da Computação — Centro Universitário do Triângulo, Uberlândia, MG.

OZSU, T. M.; VALDURIEZ, P. **Princípios de Sistemas de Bancos de Dados Distribuídos**. Rio de Janeiro: Campus, 2001.

PostgreSQL. **Sobre o PostreSQL**. Disponível em: <http://www.postgresql.org.br/sobre>. Acesso em: maio 2009.

PostgreSQL Br. **Documentação do PostgreSQL 8.2.0**. Disponível em: <http://www.postgresql.org.br/docs>. Acesso em: novembro 2009.

SACRAMENTO, A.; CALDEIRA, E.; BULHõES, F.; COSTA, F.; CARVALHO, T.; OLIVEIRA, V. de; CARDOSO, A. L. Banco de Dados em uma Perspectiva Organizacional. , Salvador, 2008.

SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. **Sistema de Bancos de Dados**. São Paulo: Makron Books, 1999.

SMITH, J. M.; BERNSTEIN, P. A.; DAYAL, U.; GOODMAN, N.; LANDERS, T. Multibase - integrating heterogeneous distributed database systems. In: AFIPS CONFERENCE PROCEEDINGS; VOL. 55 1986 NATIONAL COMPUTER CONFERENCE, 1986, Arlington, VA, USA. **Anais...** AFIPS Press, 1986. p.335–347.

SUBRAHMANIAN, V. S.; ADALI, S.; BRINK, A.; EMERY, R.; LU, J. J.; RAJPUT, A.; ROGERS, T. J.; ROSS, R.; WARD, C. HERMES: A Heterogeneous Reasoning and Mediator System. , United States, 1994.

Sun Microsystems. **JDBC 4.0 Specification**. Disponível em: <http://jcp.org/aboutJava/communityprocess/final/jsr221/index.html>. Acesso em: outubro 2009.

Sun Microsystems. **Java Standard Edition 6 API Specification**. Disponível em: <http://java.sun.com/javase/6/docs/api/>. Acesso em: novembro 2009.

WIEDERHOLD, G. Mediators in the Architecture of Future Information Systems. **The IEEE Computer Magazine**, United States, 1992.

WIEDERHOLD, G. Mediators, Concepts and Practice. In: HANDBOOK OF DATABASE SYSTEMS, 2007. **Anais...** [S.l.: s.n.], 2007.

ZANARDO, J. C. **A Técnica de Mediadores na Integração de Bancos de Dados Heterogêneos**. 1998. Monografia do Trabalho de Conclusão de Curso para obtenção do título de Bacharel em Ciência da Computação — Universidade Federal do Rio Grande do Sul, Porto Alegre, RS.

ANEXO A ALGORITMO DE DECOMPOSIÇÃO DE CONSULTAS DO PROJETO *J-INTEGRATOR*

O algoritmo abaixo foi utilizado no projeto *J-Integrator* para realizar a decomposição de consultas globais e a construção do resultado (HARA et al., 2006).

```
1  Entrada: consulta_global, ecg
2  Saída: dados_obtidos

4  //obtenção das tabelas e atributos globais utilizados
5  //na consulta global

7  tabAtrib[tabela, atrib] := tabelas_atributos(consulta_global)

9  //criação de uma tabela local para cada tabela global
10 //utilizada na consulta global

12 para cada tabelaGlobal em tabAtrib faça
13     atribGlobais := atributos(tabelaGlobal)
14     tabelaLocal := CREATE TABLE tabelaGlobal(atribGlobais)

16     //obtenção das tabelas e atributos locais necessários
17     //para popular a tabela local

19     para cada atributoGlobal em atribGlobais faça
20         fontes := obter_atributo_local(atributoGlobal, ecg)
21     fim para

23     //obtém de cada tabela local os atributos necessários
24     //para cada tabelaLocal em fontes faça
25         atribLocais := obter_atrib_locais(tabelaLocal)
26         resultado := "SELECT atribLocais FROM tabelaLocal"
```

```
27          tabelaLocal := resultado
28      fim para
29  fim para

31  //execução da consulta do usuário sobre as
32  //tabelas locais criadas

34  resFinal := executar_consulta_usuario(tabelaLocal)

36  retorna resFinal
```

ANEXO B RELAÇÃO ENTRE O PADRÃO SQL-99 E A IMPLEMENTAÇÃO NOS SGBDS MYSQL E POSTGRESQL

SQL-99	**MySQL**	**PostgreSQL**
SELECT <lista_de_atributos>	*SELECT* <lista_de_atributos>	*SELECT* <lista_de_atributos>
FROM <lista_tabelas>	*FROM* <lista_tabelas>	*FROM* <lista_tabelas>
WHERE <condicao>	*WHERE* <condicao>	*WHERE* <condicao>
<expressao_logica> *AND* <expressao_logica>	<expressao_logica> *AND* <expressao_logica>	<expressao_logica> *AND* <expressao_logica>
<expressao_logica> *OR* <expressao_logica>	<expressao_logica> *OR* <expressao_logica>	<expressao_logica> *OR* <expressao_logica>
NOT <expressao_logica>	*NOT* <expressao_logica>	*NOT* <expressao_logica>
<expressao> > <expressao>	<expressao> > <expressao>	<expressao> > <expressao>
<expressao> >= <expressao>	<expressao> >= <expressao>	<expressao> >= <expressao>
<expressao> < <expressao>	<expressao> < <expressao>	<expressao> < <expressao>
<expressao> <= <expressao>	<expressao> <= <expressao>	<expressao> <= <expressao>
<expressao> = <expressao>	<expressao> = <expressao>	<expressao> = <expressao>
<expressao> <> <expressao>	<expressao> <> <expressao>	<expressao> <> <expressao>
LIKE <cadeia_caracter>	*LIKE* <cadeia_caracter>	*LIKE* <cadeia_caracter>
BETWEEN <expressao> *AND* <expressao>	*BETWEEN* <expressao> *AND* <expressao>	*BETWEEN* <expressao> *AND* <expressao>
IN <lista_valores>	*IN* <lista_valores>	*IN* <lista_valores>
<expressao> *IS* [*NOT*] *NULL*	<expressao> *IS* [*NOT*] *NULL*	<expressao> *IS* [*NOT*] *NULL*
<expressao || <expressao>	*CONCAT*(<expressao>, <expressao>, ...)	<expressao || <expressao>
<expressao> + <expressao>	<expressao> + <expressao>	<expressao> + <expressao>
<expressao> - <expressao>	<expressao> - <expressao>	<expressao> - <expressao>
<expressao> * <expressao>	<expressao> * <expressao>	<expressao> * <expressao>
<expressao> / <expressao>	<expressao> / <expressao>	<expressao> / <expressao>
[*INNER* | (*LEFT* | *RIGHT* | *FULL*) [*OUTER*]] *JOIN*	[*INNER* | (*LEFT* | *RIGHT*) [*OUTER*]] *JOIN*	[*INNER* | (*LEFT* | *RIGHT* | *FULL*) [*OUTER*]] *JOIN*

ANEXO C DIAGRAMAS E-R DAS BASES DE DADOS PARA SIMULAÇÃO

A seguir encontram-se os diagramas Entidade-Relacionamento das bases de dados para simulação. A primeira figura ilustra as tabelas armazenadas no MySQL, enquanto a segunda mostra as tabelas armazenadas no PostgreSQL.

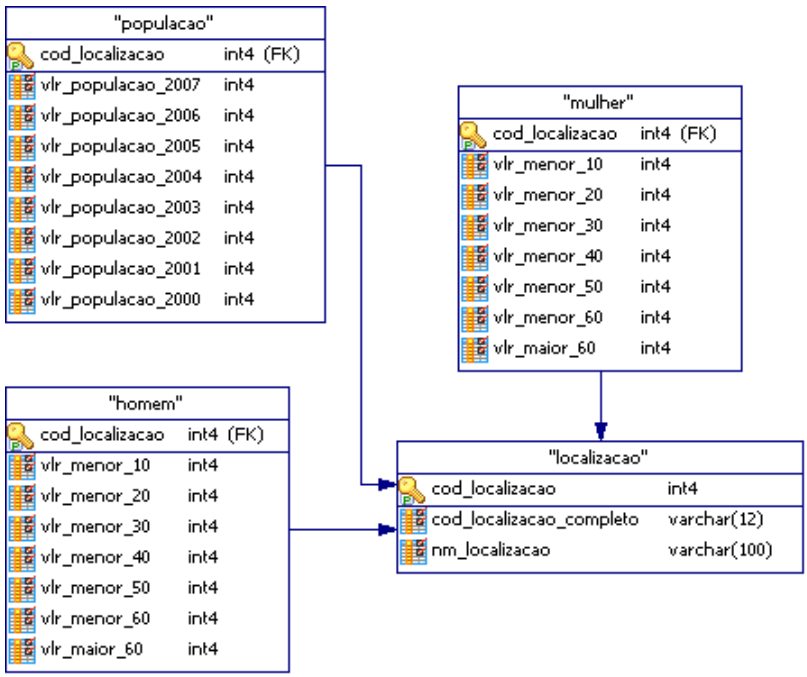

Figura C.1: Modelo Lógico do BD MySQL utilizado para simulação.

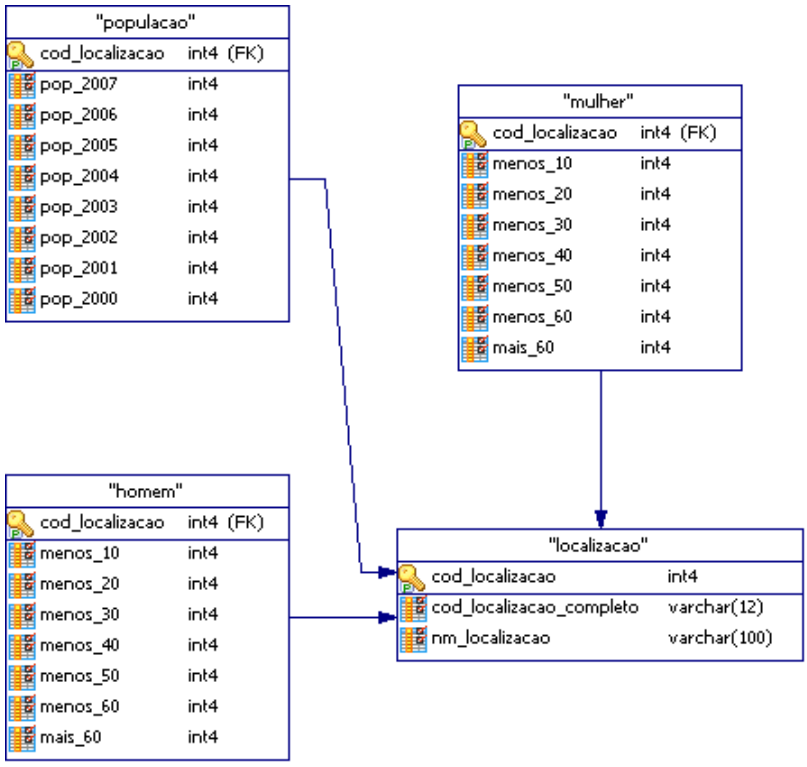

Figura C.2: Modelo Lógico do BD PostgreSQL utilizado para simulação.

ANEXO D ECG DAS BASES DE DADOS PARA SIMULAÇÃO

```
001. <?xmlversion = "1.0" encoding = "UTF-8"?>
002. <EsquemaGlobal>
003.     <Elementos>
004.         <Elemento>
005.             <CampoGlobal nome=codigo_localizacao>
006.                 <Banco nome = "postgresql">
007.                     <Tabela> localizacao </Tabela>
008.                     <Campo> cod_localizacao </Campo>
009.                 </Banco>
010.                 <Banco nome = "mysql">
011.                     <Tabela> localizacao </Tabela>
012.                     <Campo> cod_localizacao </Campo>
013.                 </Banco>
014.             </CampoGlobal>
015.         </Elemento>
016.         <Elemento>
017.             <CampoGlobal nome=codigo_local_completo>
018.                 <Banco nome = "postgresql">
019.                     <Tabela> localizacao </Tabela>
020.                     <Campo> cod_localizacao_completo </Campo>
021.                 </Banco>
022.                 <Banco nome = "mysql">
023.                     <Tabela> localizacao </Tabela>
024.                     <Campo> cod_localizacao_completo </Campo>
025.                 </Banco>
026.             </CampoGlobal>
027.         </Elemento>
028.         <Elemento>
```

```
029.            <CampoGlobal nome=nome_localizacao>
030.                <Banco nome = "postgresql">
031.                    <Tabela> localizacao </Tabela>
032.                    <Campo> nm_localizacao </Campo>
033.                </Banco>
034.                <Banco nome = "mysql">
035.                    <Tabela> localizacao </Tabela>
036.                    <Campo> nm_localizacao </Campo>
037.                </Banco>
038.            </CampoGlobal>
039.        </Elemento>
040.        <Elemento>
041.            <CampoGlobal nome=codigo_local_populacao>
042.                <Banco nome = "postgresql">
043.                    <Tabela> populacao </Tabela>
044.                    <Campo> cod_localizacao </Campo>
045.                </Banco>
046.                <Banco nome = "mysql">
047.                    <Tabela> populacao </Tabela>
048.                    <Campo> cod_localizacao </Campo>
049.                </Banco>
050.            </CampoGlobal>
051.        </Elemento>
052.        <Elemento>
053.            <CampoGlobal nome=valor_populacao_2007>
054.                <Banco nome = "postgresql">
055.                    <Tabela> populacao </Tabela>
056.                    <Campo> pop_2007 </Campo>
057.                </Banco>
058.                <Banco nome = "mysql">
059.                    <Tabela> populacao </Tabela>
060.                    <Campo> vlr_populacao_2007 </Campo>
061.                </Banco>
062.            </CampoGlobal>
063.        </Elemento>
064.        <Elemento>
065.            <CampoGlobal nome=valor_populacao_2006>
066.                <Banco nome = "postgresql">
067.                    <Tabela> populacao </Tabela>
```

```
068.                    <Campo> pop_2006 </Campo>
069.                </Banco>
070.                <Banco nome = "mysql">
071.                    <Tabela> populacao </Tabela>
072.                    <Campo> vlr_populacao_2006 </Campo>
073.                </Banco>
074.            </CampoGlobal>
075.        </Elemento>
076.        <Elemento>
077.            <CampoGlobal nome=valor_populacao_2005>
078.                <Banco nome = "postgresql">
079.                    <Tabela> populacao </Tabela>
080.                    <Campo> pop_2005 </Campo>
081.                </Banco>
082.                <Banco nome = "mysql">
083.                    <Tabela> populacao </Tabela>
084.                    <Campo> vlr_populacao_2005 </Campo>
085.                </Banco>
086.            </CampoGlobal>
087.        </Elemento>
088.        <Elemento>
089.            <CampoGlobal nome=valor_populacao_2004>
090.                <Banco nome = "postgresql">
091.                    <Tabela> populacao </Tabela>
092.                    <Campo> pop_2004 </Campo>
093.                </Banco>
094.                <Banco nome = "mysql">
095.                    <Tabela> populacao </Tabela>
096.                    <Campo> vlr_populacao_2004 </Campo>
097.                </Banco>
098.            </CampoGlobal>
099.        </Elemento>
100.        <Elemento>
101.            <CampoGlobal nome=valor_populacao_2003>
102.                <Banco nome = "postgresql">
103.                    <Tabela> populacao </Tabela>
104.                    <Campo> pop_2003 </Campo>
105.                </Banco>
106.                <Banco nome = "mysql">
```

```
107.                    <Tabela> populacao </Tabela>
108.                    <Campo> vlr_populacao_2003 </Campo>
109.                </Banco>
110.            </CampoGlobal>
111.        </Elemento>
112.        <Elemento>
113.            <CampoGlobal nome=valor_populacao_2002>
114.                <Banco nome = "postgresql">
115.                    <Tabela> populacao </Tabela>
116.                    <Campo> pop_2002 </Campo>
117.                </Banco>
118.                <Banco nome = "mysql">
119.                    <Tabela> populacao </Tabela>
120.                    <Campo> vlr_populacao_2002 </Campo>
121.                </Banco>
122.            </CampoGlobal>
123.        </Elemento>
124.        <Elemento>
125.            <CampoGlobal nome=valor_populacao_2001>
126.                <Banco nome = "postgresql">
127.                    <Tabela> populacao </Tabela>
128.                    <Campo> pop_2001 </Campo>
129.                </Banco>
130.                <Banco nome = "mysql">
131.                    <Tabela> populacao </Tabela>
132.                    <Campo> vlr_populacao_2001 </Campo>
133.                </Banco>
134.            </CampoGlobal>
135.        </Elemento>
136.        <Elemento>
137.            <CampoGlobal nome=valor_populacao_2000>
138.                <Banco nome = "postgresql">
139.                    <Tabela> populacao </Tabela>
140.                    <Campo> pop_2000 </Campo>
141.                </Banco>
142.                <Banco nome = "mysql">
143.                    <Tabela> populacao </Tabela>
144.                    <Campo> vlr_populacao_2000 </Campo>
145.                </Banco>
```

```
146.            </CampoGlobal>
147.        </Elemento>
148.        <Elemento>
149.            <CampoGlobal nome=homem_menor_10>
150.                <Banco nome = "postgresql">
151.                    <Tabela> homem </Tabela>
152.                    <Campo> menos_10 </Campo>
153.                </Banco>
154.                <Banco nome = "mysql">
155.                    <Tabela> homem </Tabela>
156.                    <Campo> vlr_menor_10 </Campo>
157.                </Banco>
158.            </CampoGlobal>
159.        </Elemento>
160.        <Elemento>
161.            <CampoGlobal nome=homem_menor_20>
162.                <Banco nome = "postgresql">
163.                    <Tabela> homem </Tabela>
164.                    <Campo> menos_20 </Campo>
165.                </Banco>
166.                <Banco nome = "mysql">
167.                    <Tabela> homem </Tabela>
168.                    <Campo> vlr_menor_20 </Campo>
169.                </Banco>
170.            </CampoGlobal>
171.        </Elemento>
172.        <Elemento>
173.            <CampoGlobal nome=homem_menor_30>
174.                <Banco nome = "postgresql">
175.                    <Tabela> homem </Tabela>
176.                    <Campo> menos_30 </Campo>
177.                </Banco>
178.                <Banco nome = "mysql">
179.                    <Tabela> homem </Tabela>
180.                    <Campo> vlr_menor_30 </Campo>
181.                </Banco>
182.            </CampoGlobal>
183.        </Elemento>
184.        <Elemento>
```

```
185.            <CampoGlobal nome=homem_menor_40>
186.                <Banco nome = "postgresql">
187.                    <Tabela> homem </Tabela>
188.                    <Campo> menos_40 </Campo>
189.                </Banco>
190.                <Banco nome = "mysql">
191.                    <Tabela> homem </Tabela>
192.                    <Campo> vlr_menor_40 </Campo>
193.                </Banco>
194.            </CampoGlobal>
195.        </Elemento>
196.        <Elemento>
197.            <CampoGlobal nome=homem_menor_50>
198.                <Banco nome = "postgresql">
199.                    <Tabela> homem </Tabela>
200.                    <Campo> menos_50 </Campo>
201.                </Banco>
202.                <Banco nome = "mysql">
203.                    <Tabela> homem </Tabela>
204.                    <Campo> vlr_menor_50 </Campo>
205.                </Banco>
206.            </CampoGlobal>
207.        </Elemento>
208.        <Elemento>
209.            <CampoGlobal nome=homem_menor_60>
210.                <Banco nome = "postgresql">
211.                    <Tabela> homem </Tabela>
212.                    <Campo> menos_60 </Campo>
213.                </Banco>
214.                <Banco nome = "mysql">
215.                    <Tabela> homem </Tabela>
216.                    <Campo> vlr_menor_60 </Campo>
217.                </Banco>
218.            </CampoGlobal>
219.        </Elemento>
220.        <Elemento>
221.            <CampoGlobal nome=homem_maior_60>
222.                <Banco nome = "postgresql">
223.                    <Tabela> homem </Tabela>
```

```
224.                    <Campo> mais_60 </Campo>
225.                </Banco>
226.                <Banco nome = "mysql">
227.                    <Tabela> homem </Tabela>
228.                    <Campo> vlr_maior_60 </Campo>
229.                </Banco>
230.            </CampoGlobal>
231.        </Elemento>
232.        <Elemento>
233.            <CampoGlobal nome=mulher_menor_10>
234.                <Banco nome = "postgresql">
235.                    <Tabela> mulher </Tabela>
236.                    <Campo> menos_10 </Campo>
237.                </Banco>
238.                <Banco nome = "mysql">
239.                    <Tabela> mulher </Tabela>
240.                    <Campo> vlr_menor_10 </Campo>
241.                </Banco>
242.            </CampoGlobal>
243.        </Elemento>
244.        <Elemento>
245.            <CampoGlobal nome=mulher_menor_20>
246.                <Banco nome = "postgresql">
247.                    <Tabela> mulher </Tabela>
248.                    <Campo> menos_20 </Campo>
249.                </Banco>
250.                <Banco nome = "mysql">
251.                    <Tabela> mulher </Tabela>
252.                    <Campo> vlr_menor_20 </Campo>
253.                </Banco>
254.            </CampoGlobal>
255.        </Elemento>
256.        <Elemento>
257.            <CampoGlobal nome=mulher_menor_30>
258.                <Banco nome = "postgresql">
259.                    <Tabela> mulher </Tabela>
260.                    <Campo> menos_30 </Campo>
261.                </Banco>
262.                <Banco nome = "mysql">
```

```
263.                    <Tabela> mulher </Tabela>
264.                    <Campo> vlr_menor_30 </Campo>
265.                </Banco>
266.            </CampoGlobal>
267.        </Elemento>
268.        <Elemento>
269.            <CampoGlobal nome=mulher_menor_40>
270.                <Banco nome = "postgresql">
271.                    <Tabela> mulher </Tabela>
272.                    <Campo> menos_40 </Campo>
273.                </Banco>
274.                <Banco nome = "mysql">
275.                    <Tabela> mulher </Tabela>
276.                    <Campo> vlr_menor_40 </Campo>
277.                </Banco>
278.            </CampoGlobal>
279.        </Elemento>
280.        <Elemento>
281.            <CampoGlobal nome=mulher_menor_50>
282.                <Banco nome = "postgresql">
283.                    <Tabela> mulher </Tabela>
284.                    <Campo> menos_50 </Campo>
285.                </Banco>
286.                <Banco nome = "mysql">
287.                    <Tabela> mulher </Tabela>
288.                    <Campo> vlr_menor_50 </Campo>
289.                </Banco>
290.            </CampoGlobal>
291.        </Elemento>
292.        <Elemento>
293.            <CampoGlobal nome=mulher_menor_60>
294.                <Banco nome = "postgresql">
295.                    <Tabela> mulher </Tabela>
296.                    <Campo> menos_60 </Campo>
297.                </Banco>
298.                <Banco nome = "mysql">
299.                    <Tabela> mulher </Tabela>
300.                    <Campo> vlr_menor_60 </Campo>
301.                </Banco>
```

```
302.             </CampoGlobal>
303.         </Elemento>
304.         <Elemento>
305.             <CampoGlobal nome=mulher_maior_60>
306.                 <Banco nome = "postgresql">
307.                     <Tabela> mulher </Tabela>
308.                     <Campo> mais_60 </Campo>
309.                 </Banco>
310.                 <Banco nome = "mysql">
311.                     <Tabela> mulher </Tabela>
312.                     <Campo> vlr_maior_60 </Campo>
313.                 </Banco>
314.             </CampoGlobal>
315.         </Elemento>
316. </Elementos>
317. </EsquemaGlobal>
```